



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁷ : G06F 17/30	A2	(11) International Publication Number: WO 00/70500 (43) International Publication Date: 23 November 2000 (23.11.00)
(21) International Application Number: PCT/US00/11946 (22) International Filing Date: 1 May 2000 (01.05.00) (30) Priority Data: 09/312,308 14 May 1999 (14.05.99) US (71) Applicant (for all designated States except US): PIVIA, INC. [US/US]; Suite 200, 10062 Miller Avenue, Cupertino, CA 95014 (US). (72) Inventors; and (75) Inventors/Applicants (for US only): MOHAN, Sudhir [US/US]; 5006 Calle de Escuela, Santa Clara, CA 95054-1431 (US). PATIL, Umesh, R. [IN/US]; 811 Milo Court, San Jose, CA 95133 (US). JORDAN, Daniel, S. [US/US]; 32 Museum Way, San Francisco, CA 94114 (US). (74) Agents: MILLIKEN, Darren, J. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).		(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG). Published <i>Without international search report and to be republished upon receipt of that report.</i>
(54) Title: CLIENT-SERVER INDEPENDENT INTERMEDIARY MECHANISM		
(57) Abstract A method and apparatus for a client-server independent intermediary mechanism is provided. The method comprises displaying a frame including a user interface of the IIM, the frame framing a destination server display area (DSDA). The method further comprises retrieving data for display from a destination server, and instrumenting the data prior to display such that future data retrieved from the destination server is displayed in the DSDA, without writing over the frame.		

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav	TM	Turkmenistan
BF	Burkina Faso	GR	Greece		Republic of Macedonia	TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's	NZ	New Zealand		
CM	Cameroon		Republic of Korea	PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

CLIENT-SERVER INDEPENDENT INTERMEDIARY MECHANISM

FIELD OF THE INVENTION

The present invention relates to client-server communication, and more specifically, to using an independent intermediary mechanism between a client and a server.

BACKGROUND

The World-Wide Web (WWW, W3, the Web) is an Internet client-server hypertext distributed information retrieval system. An extensive user community has developed on the Web since its introduction.

Figure 1 is a block diagram of a prior art client-server system. The client A 110 can access destination servers DS1-DS3 150-170. Similarly, other clients B and C, 120 130, can access the destination servers DS1-3 150-170. Each destination server may provide different services, information, or other data to the user.

On the Web everything (documents, menus, indices) is represented to the user as hypertext objects in hypertext markup language (HTML) format, or as Java, or JavaScript objects, or other data types. Hypertext links refer to other documents by their uniform resource identifiers (URIs). The client program, known as a browser, e.g. NCSA Mosaic, Netscape Navigator, runs on the user's computer and provides two basic navigation operations: to follow a link or to send a query to a server. Users access the web through these browsers.

Users often access the web from multiple locations. Users may access the web from their office, at different locations at work, at home, or on the road. Libraries and Internet cafes make web access available on a walk-in basis as well.

A user accesses a server by typing the URI of the server into the browser's address window. The browser then connects to the server corresponding to this URI. Another method of accessing a web site is by selecting the web site from the list of bookmarks. The list of bookmarks is resident in the browser in the user's computer. Thus, if a user wishes to have similar bookmarks on multiple computers, he or she must

manually copy the bookmarks and transfer them between the computers. This process is inconvenient.

Furthermore, many servers use cookies to store information about the user. This information may include the user name, password, previous interests, etc. These cookies are also stored in the user's browser. Again, this means that if the user is accessing the Internet from multiple computers, the user's cookies have to be duplicated into multiple computers. This process is inconvenient.

Many users have multiple accounts on different computer systems. For example, a user may have an account with a bank, an e-mail account, a personalized portal site account, and an account on an e-commerce server. Currently, the users must log into each of these accounts by remembering and providing his or her user name and password. For security, each of these user names and passwords should be different. Remembering different names and passwords is inconvenient to the user. Thus, a method for a simple log-in into various accounts from any computer would be advantageous.

Most clients and servers support "forms" which allow the user to enter arbitrary text as well as selecting options from customizable menus and on/off switches. As more business is transacted on the Web, forms are proliferating. The forms may include forms for requesting further information, for ordering items from the Web, for registering for a Web site, etc. However, the user generally can not get a copy of the information filled into the form. The user can either print the page when the form is filled in, generating a paper copy, or rely on the server to respond in a manner that permits the user to make a record of the information entered in to the form. A method of tracking information filled into forms would be advantageous. Furthermore, vendors may respond with an order number or other useful information. The user can keep a copy of this page, which is generally only temporarily available, by printing it, or copying down the information provided. A method of attaching this vendor response to the original order information and making both available to the user would be advantageous.

Furthermore, currently, the user has to fill out each of these forms separately. Generally, the forms request the same types of information, i.e. name, address,

telephone number, e-mail address, etc. The user has to enter all of this information for each form. This is repetitious and takes time. Additionally, if such information as credit card number or social security number is requested, the user has to pull out the credit card and copy a long string of numbers. This makes errors likely. Furthermore, the user has to verify that a Web site that requests a credit card number or similar confidential information is of the appropriate level of security for the user to feel comfortable sending the information over the Web. An improved method of filling out forms would be advantageous.

SUMMARY OF THE INVENTION

A method and apparatus of a client-server independent intermediary mechanism (IIM) is described. The method comprises displaying a frame including a user interface of the IIM (IIM frame), and a second frame framing a destination server display area (DSDA). The method further comprises retrieving data for display from a destination server, and instrumenting the data prior to display such that future data retrieved from a destination server is displayed in the DSDA, without writing over the IIM frame.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a block diagram of a prior art client-server system.

Figure 2 is a block diagram of one embodiment of the client-server system including the independent intermediary mechanism.

Figure 3A is a block diagram of one embodiment of the client-server system including multiple independent intermediary mechanisms.

Figure 3B is a block diagram of another embodiment of the client-server system including multiple independent intermediary mechanisms.

Figure 4 is a block diagram of one embodiment of the independent intermediary mechanism.

Figure 5 is a block diagram of one embodiment of the layout of the user interface of the independent intermediary mechanism.

Figure 6 is a flowchart of an overview of using the independent intermediary mechanism.

Figure 7 is a flowchart of one embodiment of the process of displaying information from a destination server through the independent intermediary mechanism.

Figure 8 illustrates one embodiment of the user interface of the independent intermediary mechanism.

Figure 9 illustrates another embodiment of the user interface of the independent intermediary mechanism.

Figure 10 is a flowchart of one embodiment of the form fill functionality.

Figure 11 is a flowchart of one embodiment of the learning process in the database.

Figure 12A is a flowchart of one embodiment of adding accounts.

Figure 12B is a flowchart of one embodiment of accessing an account through an auto-log-in feature.

Figure 13 is a flowchart of one embodiment of the transaction management functionality.

Figure 14 illustrates one embodiment of the listing of transactions.

Figure 15A is a flowchart of one embodiment of selection of a home page or a bookmark.

Figure 15B is a flowchart of one embodiment of using the bookmark functionality.

Figure 15C is a flowchart of one embodiment of using the history functionality.

Figure 16A-C are tables illustrating examples of redirecting references to DS through IIM.

Figure 17 is a table illustrating examples of making the IIM user interface frame persistent.

Figure 18 is a table illustrating examples of accessing cookies from the IIM.

Figure 19 is a table illustrating examples of preserving top frame or IIM frame integrity for DS.

DETAILED DESCRIPTION

A client-server independent intermediary mechanism is described. The independent intermediary mechanism (IIM) mediates information exchanged between a client and servers by having the client-server communication pass through the IIM. This allows the IIM to offer various services. For one embodiment, the IIM may be used to have a central web-accessible set of bookmarks. The IIM may further provide tracking of transactions on the web, providing a user-accessible transaction record. The IIM may further be used to fill in various forms automatically. The IIM may further be used to access multiple accounts, such as e-mail accounts, bank accounts, etc. with a single button. The IIM may further be used to store the user's profile, including passwords to various pages, etc. These and other uses of the IIM are described below.

Figure 2 is a block diagram of one embodiment of the client-server system including the independent intermediary mechanism. Multiple clients A-C 210, 215, 220 access multiple destination servers (DSs) 280, 285, 290, through the independent intermediary mechanism (IIM) 250. Client A 210 is described as an example. It is to be understood that multiple clients are implemented in similar ways.

Client A 210 accesses the IIM 250. For one embodiment, this occurs when the user of the client A 210 accesses the web site hosting the IIM 250. When the IIM 250 is accessed, a new client component (CC) 230 is established. The client component(s) 230, 235, 240 and the server component 260 together form the IIM 250. For one embodiment, the IIM 250 is located on a server accessed by the client A 210 through an Internet connection. For another embodiment, the IIM 250 is located within the local Intranet of client A 210. For yet another embodiment, the IIM 250 is located on the client's own computer.

For one embodiment, the client component 230 is established on the local computer of the client 210. For another embodiment, the client component 230 is on a server, or on a third computer system. The client component 230 is created in response to the client 210 connecting to the IIM 250.

The client A 210 has a connection to the server component 260 through the client component 230. For one embodiment, the client A 210 also establishes a direct connection with the server component 260. This direct connection may be used to

communicate certain information directly between the server component 260 and the client A 210. The client 210 accesses the destination servers DS1-3 280, 285, 290 through the IIM 250. For one embodiment, all of the communication between the destination server DS1 280 and the client A 210 is routed through the IIM 250. For another embodiment, certain communications are routed directly between the client A 210 and the destination server 280. For example, certain large images that do not invoke other images or other data may be routed directly in order to speed up processing.

The number of client components 230, 235, 240 depends on the number of clients 210, 215, 220 coupled to the server component 260 at any one time. For one embodiment, the server component 260 consists of multiple components that act together. A block diagram of one embodiment of the IIM 250 may be found in Figure 4, below.

Figure 3A is a block diagram of one embodiment of the client-server system including multiple independent intermediary mechanisms 350, 360. Each IIM 350, 360 is shown having a corresponding server component, 355, 365. For another embodiment, the server components 355, 365 may be located on a single server, or within a single IIM. Having server components 355, 365 coupled together may serve multiple purposes. For example, if a single IIM 350 has too many users connected to it, the IIM 350 may redirect users to a second IIM 360. For another embodiment, a user may log on to a local IIM 350, for speed reasons, and the local IIM 350 may connect to the user's "home" IIM 360 to retrieve the user's data. For yet another embodiment, the user can connect to their "home" IIM 350, which is remote, and the "home" IIM 350 may redirect the user to a local IIM 360 and send the user's data to the local IIM 360. In this way, the user's connection to the IIM 350, 360 may be optimized.

Figure 3B is a block diagram of another embodiment of the client-server system including multiple independent intermediary mechanisms. In this example, a client 310 is coupled to two IIMs 350, 360. Generally, the client 310 first connects to the first IIM 350. Then, through the user interface of the first IIM, the client 310 connects to the second IIM 360. This may be advantageous if, for example, the first IIM 350 and second IIM 360 provide different services. Thus, for example, one IIM 360 may provide additional account management features, while the other IIM 350 provides form-fill

features. By connecting to both IIMs 350, 360, in series, the user has access to the features provided by both IIMs 350, 360.

Figure 4 is a block diagram of one embodiment of the independent intermediary mechanism. The IIM 400 has three layers. The lowest layer of the IIM 400 is the core engine 410. The core engine 410 includes a server component SC and a client component CC. The Server Component, for one embodiment, is resident on the server, and handles all remote actions. The Client Component, for one embodiment, is resident on the client's system, while the client is connected to the IIM 400. For one embodiment, the client component is automatically removed from the client's system when the client disconnects from the IIM 400. The lowest layer also includes a Cookie Manager 413, for managing any cookies received from and being sent to the destination server. The use of such cookies is discussed in more detail below. Furthermore, the lowest layer may include a Activation Manager 416. The Activation Manager 416 determines if information is being transmitted by the destination server. For one embodiment, the Activation Manager 416 further determines if information is being initiated by a user's action. Information transmitted between the DS and the client is instrumented by the IIM 400, as will be described below. The Activation Manager 416 detects when the IIM 400 should review communication between the client and the DS.

The second layer is the application/UI framework layer 420. The application/UI framework layer 420 establishes the basic user interface and IIM engine. The application/UI framework layer 420 creates a persistent frame for the IIM 400. For one embodiment, the application/UI framework layer 420 further includes an instrumenting manager 425, for instrumenting data flowing from the destination server to the client, through the IIM 400. This process of instrumenting is described in more detail below.

The third layer is the applications layer. The applications layer includes multiple applications. The applications listed here are listed as an example, and are not a complete list. The applications layer, for example, may include a Navigation Manager 430. The Navigation Manager 430 permits a user to navigate from destination server to destination server using the IIM 400. The applications layer may further include a Transaction Manager 440.

The Transaction Manager 440 tracks the user's transactions, stores them, and makes them available for the user's review. Transactions are interactions in which a user submits information to a destination server, for example to order an item, ask a question, or otherwise interact with the destination server. The Transaction Manager 440 tracks the data submitted by the user, and any response from the destination server, and permits the user to access this information.

The Account Manager 450 permits the user to log into a variety of accounts, from e-mail to stock trading accounts, using a single click. The Account Manager 450 further permits the user to add accounts to this list. The Form Manager 460 permits the user to fill out forms encountered on destination servers via a single click. This is extremely useful for users that transact business on the web, and often fill out identical forms many times. The Profile Manager 470 is the database of the user's personal information. This information may be edited by the user, and is used to fill in forms via the form manager 460. The Database Manager 480 manages the various databases of the IIM 400.

The Bookmark Manager 490 permits the user to manage bookmarks maintained within the IIM 400. Having bookmarks, URIs of pages the user wishes to save, available in the IIM 400 permits the user to access his or her bookmark list from any computer.

The History Manager 495 permits the user to manipulate the history list of sites the user has previously visited. For one embodiment, the user can change the permanence of the history lists, for another embodiment, the user can delete certain sites from the history list.

The Homepage Manager 497 permits the user to set a homepage that is displayed when the user initially connects to the server providing the IIM 400.

As can be seen, the IIM provides multiple functionalities. A single IIM 400 may include all of the functionalities described above, additional functionalities, or some subset of these functionalities. The IIM's functionality may be extended with additional features.

Figure 5 is a block diagram of one embodiment of the layout of the user interface of the independent intermediary mechanism. The client browser application window 510 is displayed by a browser, such as Netscape Navigator or Microsoft Internet Explorer. The client side display area (CSDA) 520 is the display area available in the

browser application window 510. Most browsers have a toolbar and other displays within the browser application window 510. For one embodiment, the IIM is designed to minimize the area of the browser application window that is not the CSDA 520.

The CSDA 520 includes a toolbar frame 530. Although the tool bar frame 530 is referred to as a frame, for one embodiment, the tool bar frame 530 may be implemented in a non-frame format. For one embodiment, the tool bar may be implemented as a separate window. For another embodiment, the tool bar may be implemented as part of the display, not as a frame.

The CSDA 520 further includes a destination server display area (DSDA) 540. The DSDA 540 is the area in which all information from destination servers is presented.

The CSDA 520 further includes a communications frame 550. The communications frame 550 is for communication between the client side and server side of the IIM. Generally, the communications frame 550 is hidden from the user's view. Thus, the user would not see the communication between the client component and the server component.

Figure 6 is a flowchart of an overview of using the independent intermediary mechanism. At block 610, the user connects to the IIM through the client browser. For one embodiment, this is done by typing the address of the IIM into the address window of the browser. For one embodiment, the IIM may be the preset homepage of the user, or a bookmark in the client browser.

At block 615, the user connects to a destination server (DS) through the IIM. For one embodiment, this is done by typing the address of the destination server into the address window of the IIM. For another embodiment, the user may select an address from a history list of previously visited sites, from a bookmark list in the IIM, or the destination server may be a preset homepage in the IIM. The IIM records the DS in the history database. The history database tracks the web sites that the user has visited in the past. Such a history database may be useful to permit backtracking, or to visit previously visited sites. For one embodiment, this history database is maintained for a fixed duration of time, or a user preset period of time. For another embodiment, the history database is maintained indefinitely.

At block 620, the process changes the reference to DS to go through the IIM and load the information from the DS in the DSDA, maintaining the IIM frame. This is described in more detail below.

At block 625, the IIM determines whether the user submitted information to the destination server. For one embodiment, the actual test is whether information that is "sensitive" or "of interest" is submitted to the DS. For example, if a user selected a radio button for the next display, the response would be "no" even though some information was submitted. For one embodiment, the answer to this query is yes only if information that is in the user's profile is submitted. For one embodiment, the answer to this query is provided by the user through the user interface. If the answer is yes, the process continues to block 630.

At block 630, the user's communication with the DS is recorded in the user's transaction database. For example, if the user ordered an item from a destination server site, the form that was filled in by the user, including all of the information filled in, would be recorded in the transaction database. This transaction database is available to the user. The process then continues to block 635. If, at block 625, the answer was no, the process continues directly to block 635.

At block 635, the IIM forwards the communication, i.e. the information submitted by the user, to the DS. This communication includes relevant cookies. A cookie is a packet of information sent by a destination server to a browser and then sent back by the browser each time it accesses that server. Cookies can contain any arbitrary information the server chooses and they are used to maintain state between otherwise stateless transactions. Generally, cookies are maintained in a user's browser. However, for one embodiment, the IIM maintains the user's cookies. This permits a user to log into a site, and have the appropriate cookies available, no matter from what web client device or client browser the user accesses the site.

At block 640, the process determines whether the destination server responded to the user's submission of information. For one embodiment, some destination servers respond, with a thank you page, other data pertaining to order number, shipping code, delivery date, etc., when information is submitted to them. If the destination server responds at block 640, the process continues to block 645.

At block 645, the DS's response is recorded in the user's transaction database, and associated with the user's submitted information. Thus, when the user reviews the transaction, he or she can review the entire transaction, including the DS's response.

At block 650, the IIM instruments the DS's response, stores any cookies returned by the DS, and forwards the response to the client browser. One embodiment of this process is illustrated in more detail in Figure 7, below. Tables of some results of the process of instrumenting are illustrated in Figures 16A-C, and Figures 17-19.

At block 655, the process tests whether the user continues to browse through the IIM. The user continues to browse, the process returns to block 615. Otherwise, the process ends at block 660.

Figure 7 is a flowchart of one embodiment of the process of instrumenting data displayed from a destination server through the independent intermediary mechanism. For one embodiment, Figure 7 is a more detailed flowchart of block 650, in Figure 6. At block 705, the IIM receives a communication from the DS. For one embodiment, this occurs in response to a user contacting a DS through the IIM.

At block 710, the process tests whether there is a cookie or multiple cookies associated with the communication. Cookies may be sent by the DS to the client, to be stored on the client browser. If a cookie is associated with the communication, the process continues to block 715. At block 715, the IIM cookie database is updated with the new cookie. For one embodiment, cookies sent by the DS to the client browser are handled through the IIM. Thus, the IIM would store all of the cookies for a DS, and give the DS its cookies. This is advantageous because it permits a user to access a DS from any computer, and all of the user's cookies are immediately available through the IIM. The process then continues to block 725. If no cookies were associated with the communication, the process continues directly to block 725.

At block 725, the process parses the code to find the next keyword. For one embodiment, keywords are tags in HTML, or known keywords in Java or JavaScript. Figures 16-19 illustrate some examples of keywords that may trigger this process. For another embodiment, keywords may be any triggering signal that indicates that an action may be performed.

At block 730, the process tests whether a keyword was found. If no keyword was found, the process continues to block 735, and ends. If the communication has no remaining keywords, the document has been fully instrumented, and is ready for display to the user. For one embodiment, certain communications may have no keywords at all. In that case, this process would end after a single pass. For yet another embodiment, under some circumstances, the process may ignore certain keywords. Certain references are not altered in the communication. For example, references that call static images, images that do not communicate information to the user and do not have embedded references, may be of no interest. For example, if the keyword calls a large passive figure with multiple components, the process may ignore the entire figure, by tagging figure related communications, and exit out of this process even if keywords remain. By altering only those references that are of interest, the process may be sped up. If a keyword was found, the process continues to block 740.

At block 740, the process tests whether the keyword is an attempt to access a cookie from the cookie database. If the keyword is an attempt to access a cookie, the process continues to block 745. At block 745, the access attempt is changed to fetch the cookie from the IIM's cookie database. Some examples of this process are provided in Figure 18. For one embodiment, the IIM's cookie database may access the client browser's cookie database in order to determine whether there are additional cookies on the client browser. For one embodiment, the IIM can, with the user's permission, copy cookies from the browser cookie database to the IIM. This simplifies moving from direct access of a DS to accessing a DS through the IIM. The process then continues to block 750.

If the keyword is not an attempt to access a cookie, the process continues directly to block 750.

At block 750, the process tests whether the keyword is an attempt to access the top frame or IIM frame. If the keyword is an attempt to access the top frame or IIM frame, the process continues to block 755. At block 755, the access attempt is changed to access the top area of the destination server display area (DSDA). Some examples of this process are provided in Figure 17. The process then continues to block 760.

If the keyword is not an attempt to access the top of IIM frame, the process continues directly to block 760.

At block 760, the process tests whether the keyword is a reference to the destination server. If the keyword is a reference to the destination server, the process continues to block 765. At block 765, the reference is changed to be fetched through the IIM. Some examples of this process are provided in Figure 16A-C. The process then continues to block 770.

If the keyword is not a reference to the destination server, the process continues directly to block 770.

At block 770, the process tests whether the keyword is an attempt to access data from the top frame or IIM frame. If the keyword is an attempt to access data from the top frame or IIM frame, the process continues to block 775. At block 775, the access attempt is changed to fetch data from the topmost frame of the DSDA. Some examples of this process are provided in Figure 19. The process then returns to block 725, and parses to find the next keyword.

For one embodiment, the above process may be triggered by a user. For example, a user may select a link, activate a JavaScript function, or otherwise initiate communication between the destination server and the client. The same process may occur in response to a cookie being sent or received, or a keyword being found as described above with respect to Figure 7.

Figure 8 illustrates one embodiment of the user interface of the independent intermediary mechanism. The user interface includes a browser toolbar 805. For one embodiment, the IIM may configure the browser such that the browser toolbar area 805 is not displayed when the IIM is active. The display area 810 of the browser includes the IIM toolbar 820, a hidden communications frame 815, and the destination server display area 845.

The IIM toolbar 820 includes the known browser controls 825, such as back, forward, refresh, stop, etc. Additional browser controls 825 may be added. The toolbar 820 further includes an address entry control 830, where a user can type a destination server address in order to access the DS.

The IIM toolbar 820 may further include buttons, or other selection mechanisms that permit a user to configure and use the IIM. The buttons may include Home, selecting a user's preset homepage, etc. The homepage is preset using the Set Home button 852. The buttons may further include the Mall button, giving one-button access to shopping. The buttons may further include Tags 860, displaying a list of a user's bookmarks. Bookmarks are added by selecting the Tag Address while visiting a web site, or by selecting the Tag Address button 862, and typing the address of a location to be bookmarked.

The buttons may further include Accounts 865, permitting single-button log-on to a variety of accounts. These accounts are added with the Add Account button 867, as will be described below.

The buttons may also include a Transactions button 870, that permits a user to review his or her transactions. This is illustrated in the destination server display area 845 of Figure 8. The Profile button 875 permits the user to enter his or her personal data. The Fill-Form button 880 permits the user to fill in a form using the personal data from the user's profile or by using information submitted previously using the same form. If a form is displayed on the destination server display area 845, and the user selects the fill-form button 880, the form is automatically filled in with the user's information. The Clear Form button 882 permits a user to remove the information filled into a form. This provides an additional level of security to the user.

The Admin button 885 provides access to account administration services. For one embodiment, the Admin button 885 is only available to those users who are authorized administrators. For one embodiment, the Admin button 885 is only displayed if the user is authorized to access account administration services.

The toolbar 820 further includes a Bye button 890, which logs off the user from the IIM. The toolbar 820 illustrated is exemplary. The content and organization of the buttons on the toolbar 820 may be changed without changing the invention.

Figure 9 illustrates another embodiment of the user interface of the independent intermediary mechanism. As can be seen, the user interface may be flexibly implemented. Certain features may be provided by one interface and not provided by another. Furthermore, the look and feel of the user interface may be altered. The user

may, for example, access all of the IIM features through pull-down menus, such as the pull-down menu 935, or radio buttons instead of buttons. One skilled in the art understands other types of user interface changes that may be made without departing from the broader spirit and scope of the invention as set forth in the appended claims.

Figure 10 is a flowchart of one embodiment of the form fill functionality. At block 1010, a document with a form is displayed. For one embodiment, this is a result of a user accessing a destination server location that includes a form. This form may be an order form, an information request form, or any other form that may be encountered on the Web.

At block 1015, the user requests the form-fill function through the IIM user interface. For one embodiment, the user presses the form-fill button. For another embodiment, the form fill may be automated. For yet another embodiment, the user can select whether the form fill function is automatically engaged.

At block 1020, the process determines whether the form is in the user's transaction database. The user's transaction database has records of previously accessed and filled-in forms for the particular user. The transaction database may maintain such records for a limited time, or the user may delete transaction records. Thus, merely because a user has been to a particular site previously may not mean that the form is in the user's transaction database. If the form is in the user's transaction database, the process continues to block 1040, otherwise, the process continues to block 1025.

At block 1025, the process determines whether the form is in the form database. The form database is maintained by the IIM and includes "known" forms. Such known forms have associations between form control identifiers in the form and profile items. Thus, for example, a form control identifier that is labeled "name" may have a link to the "First Name Last Name" item in the user profile. If the form is known, the process continues to block 1030. At block 1030, the form control identifiers in the form are filled in from the user profile. The process then returns to block 1055.

If the form is not known, the process continues to block 1035. At block 1035, the form controls are identified, based on the name of each control. Each control name is associated with entries in the user profile. The process then continues to block 1030,

and the data is filled into the form from the user profile. For one embodiment, block 1035 is skipped. This type of "guessing" may be user enabled, or may be only attempted for forms that are similar to known forms.

At block 1020, if the form was found in the user's transaction database, the process continued to block 1040. At block 1040, the process tests whether any data in the user profile has been changed since the transaction in the transaction database was recorded. Transaction records are dated, as are changes to the user profile. A user profile may be changed by the user, for example, to change a credit card expiration date, number, or home address. If a user profile change of a relevant field is dated after the transaction record date, the process continues to block 1045, otherwise, the process continues directly to block 1050.

At block 1045, the changed information is filled in from the user profile. In this way, the user only had to update his or her records once, in the profile, and that change is carried through the IIM. For one embodiment, this step may be skipped. For another embodiment, this step may be user enabled.

At block 1050, the remaining form control identifiers in the form are filled with data from the transaction database. The process then continues to block 1055.

At block 1055, the filled-in form is displayed to the user, and the user is permitted to edit the data in the form. The user, for example, may not wish to provide certain data to a destination server. The user may chose to erase such data. Alternatively, the form may request data that is not found in the user's profile. The user may chose to fill in such data.

At block 1060, the user submits the form to the destination server. For one embodiment, the IIM stores the information submitted to the server in the user's transaction database. This is illustrated in Figure 13 below. At block 1065, the process ends. For one embodiment, the user may optionally select whether to use the user profile, transaction database, or both, and in what order, for form fill functions.

Figure 11 is a flowchart of one embodiment of the learning process in the database. At block 1110, a privileged user connects to the IIM. For one embodiment, this privileged user is an employee of the group maintaining the IIM. For another

embodiment, this "user" is an artificial intelligence unit that is used to identify forms, as will be described below. Such intelligent recognition programs are known in the art.

At block 1120, the privileged user accesses a destination server page with a form through the IIM. At block 1130, the IIM displays a user interface for cataloguing the form.

At block 1140, the user maps each form control to an element in the user profile object. The user profile is set up to contain a large number of possible data elements. Each form control should have a corresponding profile element. If no profile element is found for a form control, that form control may be tagged as "form specific." For one embodiment, multiple elements in the user profile may be associated with a single form control, or vice versa.

At block 1150, other information about the form is added. This information may include such information as the address of the form, whether the connection with the destination server that serves the form is a secure connection, whether the form is of a particular classification, etc.

At block 1160, the user submits the information to the IIM.

At block 1170, the IIM updates the form identification and form description in the form database to include the information added by the user. For one embodiment, the updating is a periodic batch updating. For one embodiment, a single central form database is maintained. In that instance, the IIM's updating may include sending the new form to other IIMs. Alternatively, each IIM may maintain its own separate form database. For yet another embodiment, an IIM may have a central form database, and a separate internal form database. This may be useful, for example, for an IIM implemented within a company which has the general form database for pages accessed outside the company, and a separate internal database for internal web page forms.

At block 1180, the process ends. Of course, the privileged user may enter multiple entries, and may start the process again at block 1120.

Figure 12A is a flowchart of one embodiment of adding accounts. At block 1210, the user connects to the IIM through a client browser. At block 1220, the user accesses a destination server through the IIM. For one embodiment, the user accesses the account

log-in page of the DS. This may be, for example, the account log-in page of the user's bank, of a portal, or of any other DS.

At block 1230, the user requests to add the account to the user's account database. Each user may have an account database, which includes a list of accounts the user can access with a single click.

At block 1235, the process determines whether the user has submitted log-in information to the account log-in page. If the user has not submitted the information, the process continues to block 1240, and the user is prompted to complete the log-in process. For one embodiment, if the account log-in process includes multiple pages, the user may indicate the end of the log-in process by pressing a certain key, or through other means. The process then continues to block 1245. If the user has submitted all of the log-in information, the process continues to block 1245 directly.

At block 1245, the account entry is added to the user's account database. The account log-in information and data of account entry creation are recorded. For one embodiment, further information may be recorded. For yet another embodiment, only the user's log-in procedure is recorded.

At block 1250, the account information is submitted to the DS for login. At block 1255, the process ends.

Figure 12B is a flowchart of one embodiment of accessing an account through an auto-log-in feature. At block 1260, the user connects to the IIM. At block 1265, the user accesses the account auto-log-in feature using the IIM user interface. For one embodiment, this is done by the user pushing the account button.

At block 1270, the user selects an account to log into. For one embodiment, the user may have multiple accounts. In that instance, the IIM displays the accounts that the user has. For another embodiment, if the user only has a single account, that account is automatically selected when the user accesses the auto-log-in feature.

At block 1275, the IIM retrieves login information from the user's account database. As discussed above, the user's previous account log-in is monitored and recorded. This information is retrieved at block 1275.

At block 1280, the IIM sends the log-in information to the appropriate destination server to log-in the user. The account information includes the address of the DS. The IIM accesses the DS as a client, and sends the user's information.

At block 1285, the IIM instruments the DS's response and sends it to the user's browser for display. As discussed above, the response is instrumented such that references of interest are routed through the IIM. The user can now use the account, as usual. At block 1290, the process ends.

Figure 13 is a flowchart of one embodiment of the transaction management functionality. At block 1310, the user connects to the IIM.

At block 1320, the user transmits information in a form to the destination server. For one embodiment, the user first accesses a destination server page including a form through the IIM. This form may be an order form, an e-mail form, or any other type of form. The user then fills in the form and submits it to the DS. For one embodiment, the user may use the form-fill method described above to fill-in the form.

At block 1330, the process determines whether the user sent the user's e-mail address to the DS. The user may submit his or her e-mail address so the DS can send responses directly to the user's e-mail. For example, certain systems may send confirmation e-mails or alert notices to the user via e-mail. If the user submitted his or her e-mail address, the process continues to block 1340. Otherwise, the process continues directly to block 1350.

At block 1340, the e-mail address submitted to the DS is altered. Specifically, the e-mail address is bifurcated, generated two e-mails. The first e-mail goes to the user's e-mail address, as entered. The second e-mail goes to the IIM. The second e-mail includes in its address the IIM and the transaction tag that identifies the transaction number to which the e-mail belongs. This allows the IIM to handle the e-mail. The process then returns to block 1350.

At block 1350, the IIM records a transaction in the user's transaction database and associates the submitted information with the transaction. The transaction, for one embodiment, has a transaction number.

At block 1360, the IIM determines whether there is a response from the DS. If there is a response, the process continues to block 1370. Otherwise, the process continues directly to block 1380.

At block 1370, the IIM records the response from the DS in the user's transaction database. For one embodiment, the destination server may respond to the user. This response is associated with the transaction record. In this way, the user may review the transaction record, including the response.

At block 1380, further information is recorded about the transaction. For one embodiment, this information may include the date and time of the transaction, and other information.

At block 1390, any notes, data, or e-mails received with the transaction tag are attached to the transaction. This may occur at any time, while the transaction is being recorded, or after that. The user may attach any data to the transaction, and the IIM may automatically attach any e-mails received with the transaction tag.

At block 1395, the process ends.

Figure 14 illustrates one embodiment of the listing of transactions. The transaction list 1410 may be sorted by date, using a menu 1425. The transactions may also be sorted by type 1435. For one embodiment, alternative methods of searching transactions may also be implemented. For example, a user may search the transaction records for purchases from a certain destination server.

Each transaction record may include one or more of the following: date 1420, transaction type 1430, and description 1440 of the transaction. The record may further include the place 1450, the location from where the transaction was recorded. The user may add and edit additional notes 1460. Furthermore, the user may also add attachments 1415 to the transaction record. For example, the user may attach e-mails, documents, video, or other types of data. For one embodiment, e-mails may be redirected through the IIM and automatically attached to the transaction.

The vendor response 1470 is also recorded. The information the user provided 1480 during the transaction is also included in the transaction record. The transaction may further include the information whether the transaction belongs to one of the

accounts 1490 in the user's account database. The user is permitted to delete selected transaction records using a delete button 1465.

Figure 15A is a flowchart of one embodiment of selection of a home page. The user connects to the IIM at block 1505.

At block 1510, the user accesses a destination server page through the IIM. At block 1515, the process determines which option the user is selecting.

If the user is selecting the add bookmark option, the process continues to block 1525. At block 1525, the address of the page is added to the user's bookmark database. This database is accessible to the user, to permit the user to access various web sites without typing the address of the site. The process then continues to block 1530, and ends.

If the user selected the set home page option at block 1515, the address of the page is set as the user's homepage. The user's homepage is called up when the user initially connects to the IIM. For one embodiment, the homepage is preset. For another embodiment, the user may not alter the homepage, and the homepage is customizable but includes advertising. The process then continues to block 1530, and ends.

Figure 15B is a flowchart of one embodiment of using the bookmark functionality. At block 1535, the user connects to the IIM. At block 1540, the user requests access to the user's bookmarks through the IIM user interface. For one embodiment, the user requests the bookmarks by pressing the "Tags" button on the user interface.

At block 1545, the IIM generates a bookmark list from the user's bookmark database, and sends the list to the client browser to display. For one embodiment, the bookmark list is displayed in the destination server display area. For another embodiment, the bookmark list is displayed in a separate window, or a separate frame.

At block 1550, the user selects a bookmark to access a destination server page.

At block 1555, the IIM fetches the page address corresponding to the selected bookmark from the bookmark database. The bookmark database includes the actual address of the bookmark.

At block 1560, the destination server page is fetched by the IIM. The data from the destination server is instrumented and is sent to the client browser for display. In

this way, the user can access bookmarks stored in the IIM's bookmark database. The process then continues to block 1565, and ends.

Figure 15C is a flowchart of one embodiment of using the history functionality. At block 1570, the user connects to the IIM.

At block 1575, the user requests access to the history list through the IIM user interface. The history list includes the sites the user previously visited. For one embodiment, the history list is maintained for only a period of time, such as thirty days. For another embodiment, the history list is maintained indefinitely, and may be purged by the user.

At block 1580, the IIM generates a history list from the user's history database, and sends the history list to the client browser for display. For one embodiment, the history list is displayed in the destination server display area. For another embodiment, the history list is displayed in a separate window, or a separate frame

At block 1582, the user selects a list entry to access the destination server page. At block 1585, the IIM fetches the page address from the history database. The page address is referenced through the IIM.

At block 1590, the IIM fetches the destination server page, instruments the communication, and sends the data to the client browser for display. At block 1595, the process ends. In this way, the IIM permits a user to access a variety of services through the IIM.

Figures 16A-C show sample alterations of references from the destination server by the IIM. Figures 16A-C illustrate changes to HTML, HTTP protocol, JavaScript, and Java. For one embodiment, this technique may be expanded to new languages and other types of interfaces. The data that is normally communicated directly between a Destination Server (DS) and client browser is altered by the IIM, as shown by Figures 16A-C. For one embodiment, some data may be transmitted directly between the DS and the client browser, without passing through the IIM.

For one embodiment, the IIM performs a subset of the message modifications required for redirection and downloads the client component to the client's browser, which performs the remaining subset of message modifications on the client machine.

Together these two subsets of message modifications provide a complete solution for using an independent intermediary mechanism between a client and a server.

The modification of HTTP communication messages for redirection occurs on both the IIM and the client browser using the client component. The points at which the message modifications occur are called "HTTP control points".

Figures 16A-C illustrate examples of HTTP control points that occur on the client browser and the IIM. For HTTP message documents, description of modification code covers the three programming languages that are most widely used today for HTTP communication: HTML, JavaScript and Java. For another embodiment, the IIM utility may be broadened to include HTTP control points in other programming languages used for HTTP message documents. For one embodiment, the protocol modified in the messages is defined by the HTTP specification standard. One skilled in the art would understand how to expand the technique described to different programming languages or message protocols.

Figure 17 is a table illustrating examples of making the IIM user interface frame persistent. The IIM prevents DS's from overwriting the user interface of the IIM. This permits the user to access the IIM regardless of what DS he or she is accessing.

Figure 18 is a table illustrating examples of accessing cookies from the IIM. Generally, the destination server and destination server data on the client system access the cookie cache on the client's computer system. The IIM modifies the access mechanisms to access cookies from the IIMs cookie database.

Figure 19 is a table illustrating examples of preserving top frame or IIM frame integrity for DS. Objects are often hung from the top frame of the client browser. The IIM changes the references to the top frame to create or access these objects to references to the top frame of DSDA. In this way, the objects are appropriately handled.

Figures 16-19 list some sample alterations resulting from the code instrumenting described above. Alternative methods of altering the code may be used. One skilled in the art knows how to implement different changes.

In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader

spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method of accessing data through an independent intermediary mechanism (IIM), the method comprising:
 - displaying a frame including a user interface of the IIM, the frame framing a destination server display area (DSDA);
 - retrieving destination server data (DS data) for display from a destination server;
 - instrumenting the DS data prior to display such that future data retrieved from the destination server is displayed in the DSDA, without writing over the frame displaying the user interface of the IIM.
2. The method of claim 1, further comprising:
 - altering requests for cookies such that cookies relevant to the destination server are accessed from the IIM.
3. The method of claim 1, further comprising:
 - determining if a user's portion of the IIM includes a cookie for the destination server, and serving the cookie to the destination server and to the DS data, if the user's portion includes the cookie.
4. The method of claim 3, further comprising determining if a browser includes the cookie, and if the browser includes the cookie:
 - serving the cookie to the destination server and the DS data; and
 - saving the cookie in the user's portion of the IIM.
5. The method of claim 2, wherein cookies received from the destination server or created by the DS data are stored in a user's portion of the IIM.
6. The method of claim 1, wherein at least one reference in the DS data to other DS data is redirected through the IIM.
7. The method of claim 1, wherein the step of instrumenting data prior to display comprises replacing a reference to a top frame or IIM frame with a reference to a top of the DSDA.
8. The method of claim 7, wherein said step of replacing comprises,
 - in HTML, determining if a value of a Target attribute is "_top", and changing the value to represent a topmost area of the DSDA.

9. The method of claim 7, wherein said step of replacing comprises, in Java, determining if a value of a Target attribute is “_top”, and changing the value to represent a topmost area of the DSDA.
10. The method of claim 7, wherein said step of replacing comprises, in JavaScript, replacing the reference to “top” with a reference to a topmost area of the DSDA.
11. The method of claim 6, wherein for predefined JavaScript, HTML and other code, the step of instrumenting is performed on a server side of the IIM, and wherein for dynamically generated code, the step of instrumenting is performed on a client side of the IIM.
12. The method of claim 1, wherein the step of altering data prior to display comprises replacing the DS data references to a reference through the IIM.
13. The method of claim 12, wherein only selected references are routed through the IIM.
14. The method of claim 12, wherein said step of replacing comprises altering a language of the reference such that any parameter which when set causes a document to be fetched from the destination server causes the document to be fetched through the IIM.
15. The method of claim 1, wherein links and references invoked by a user’s selection are altered when the user selects the reference.
16. A method of accessing data through an independent intermediary mechanism (IIM), the method comprising:
 - retrieving destination server data (DS data) for display from a destination server;
 - instrumenting the DS data such that future data retrieved from the destination server is retrieved through the IIM.
17. A method of accessing data through an independent intermediary mechanism (IIM), the method comprising:
 - retrieving destination server data (DS data) for display from a destination server;
 - altering a request for a cookie from the destination server or the DS data, such that the cookie relevant to the destination server is accessed from the IIM; and
 - storing and updating the cookie in the IIM cookie database.

18. An independent intermediary mechanism (IIM) comprising:
a core engine retrieving destination server data (DS data) for display from a destination server;
a user interface framework for maintaining a frame including the IIM user interface on a client browser as the client browser accesses different destination servers;
19. The IIM of claim 18, further comprising:
a cookie database;
a cookie modification engine that alters a request for a cookie from the destination server or the DS data, such that the cookie relevant to the destination server is accessed from the IIM cookie database; and
the cookie modification engine further for maintaining and updating the cookie.
20. The IIM of claim 18, further comprising:
a data modification engine for instrumenting the DS data such that future data retrieved from the destination server is retrieved through the IIM.
21. A communications mechanism comprising:
a first independent intermediary mechanism (IIM) displaying a frame including a user interface of the IIM, the frame framing a destination server display area (DSDA);
the first IIM retrieving destination server data (DS data) for display from a destination server and instrumenting the DS data prior to, the first IIM further for providing services to the user.

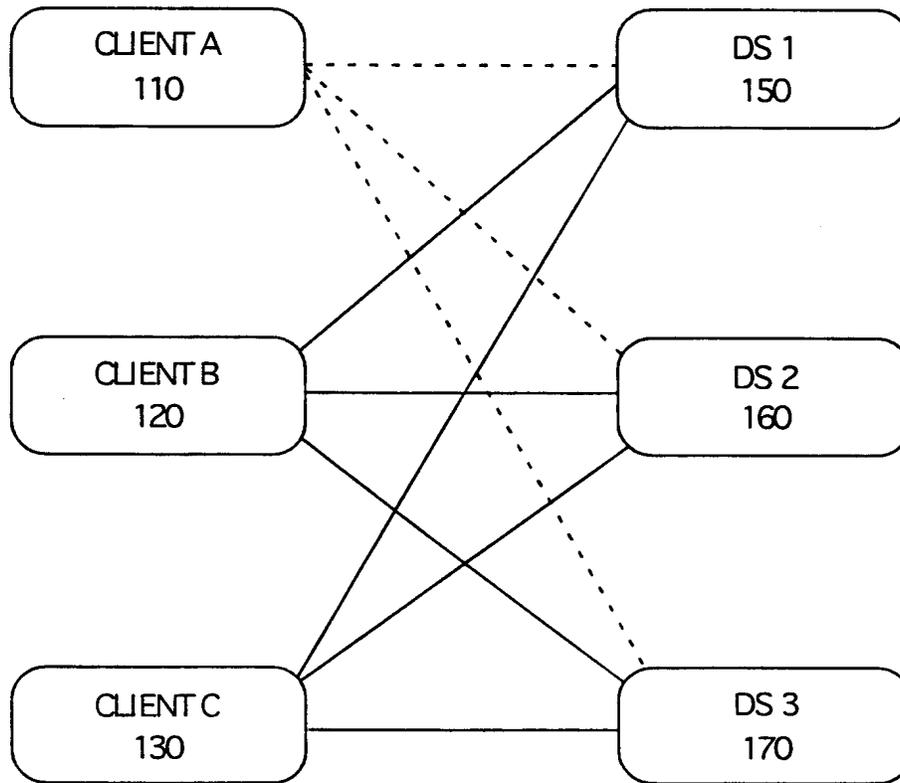


FIG. 1 (PRIOR ART)

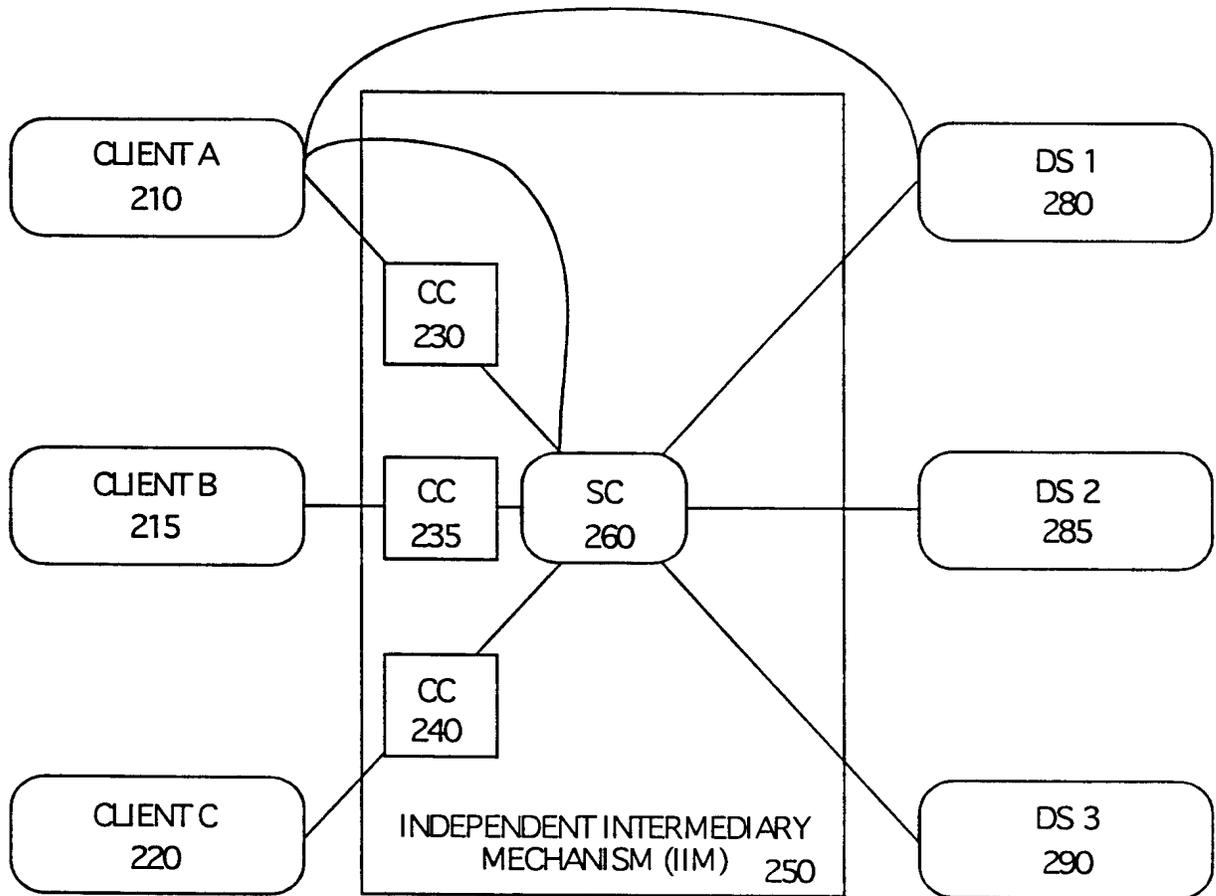


FIG. 2

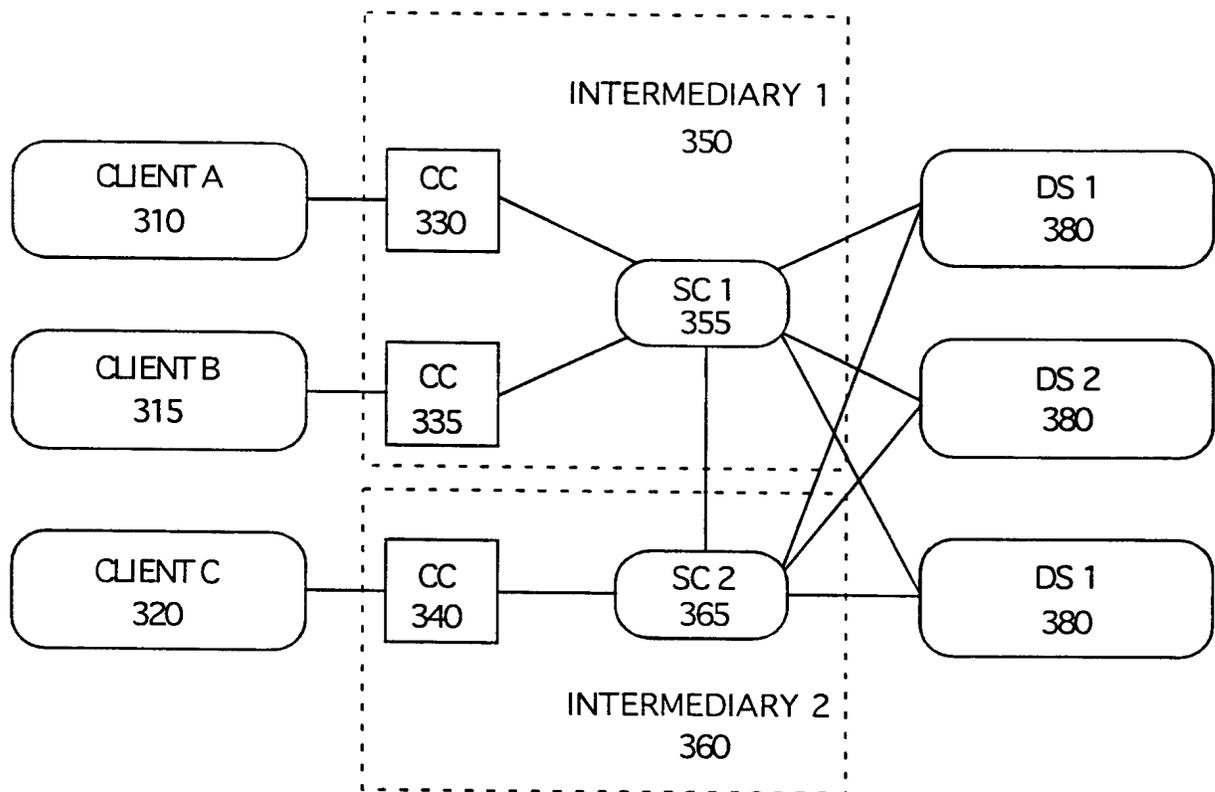


FIG. 3A

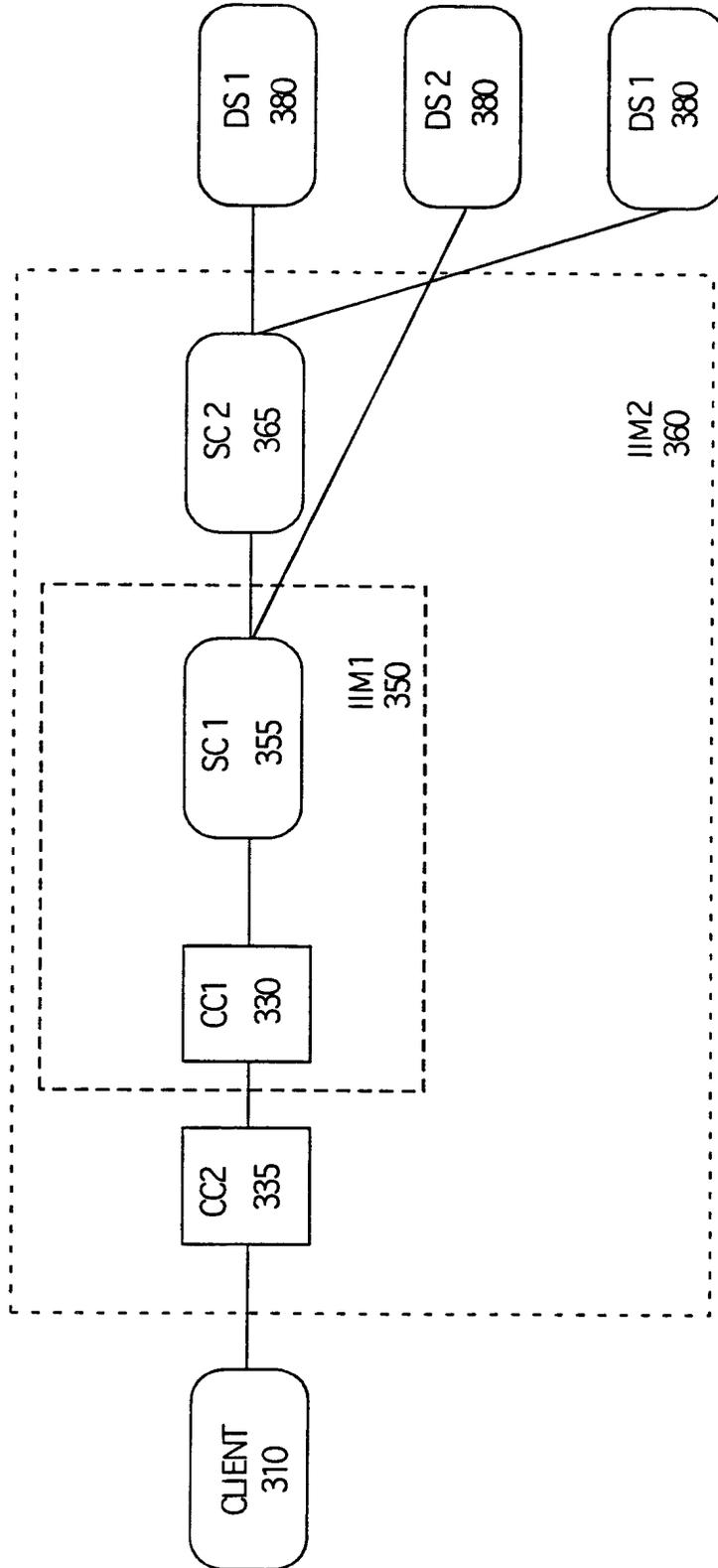


FIG. 3B

INDEPENDENT
INTERMEDIARY
MECHANISM
(IIM)
400

→

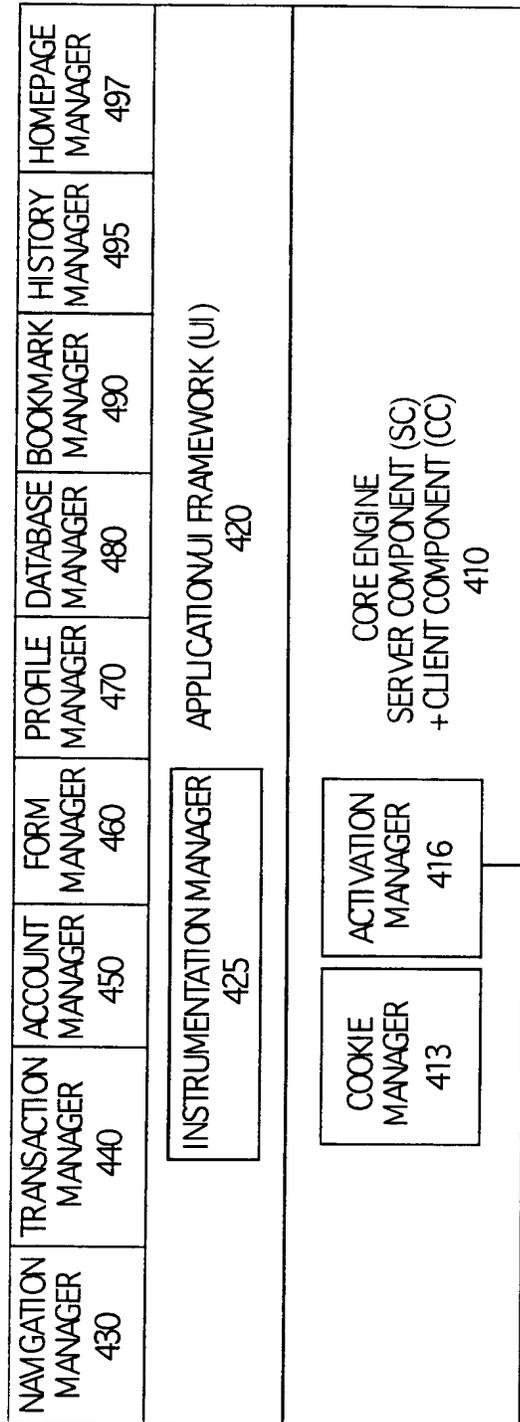


FIG. 4

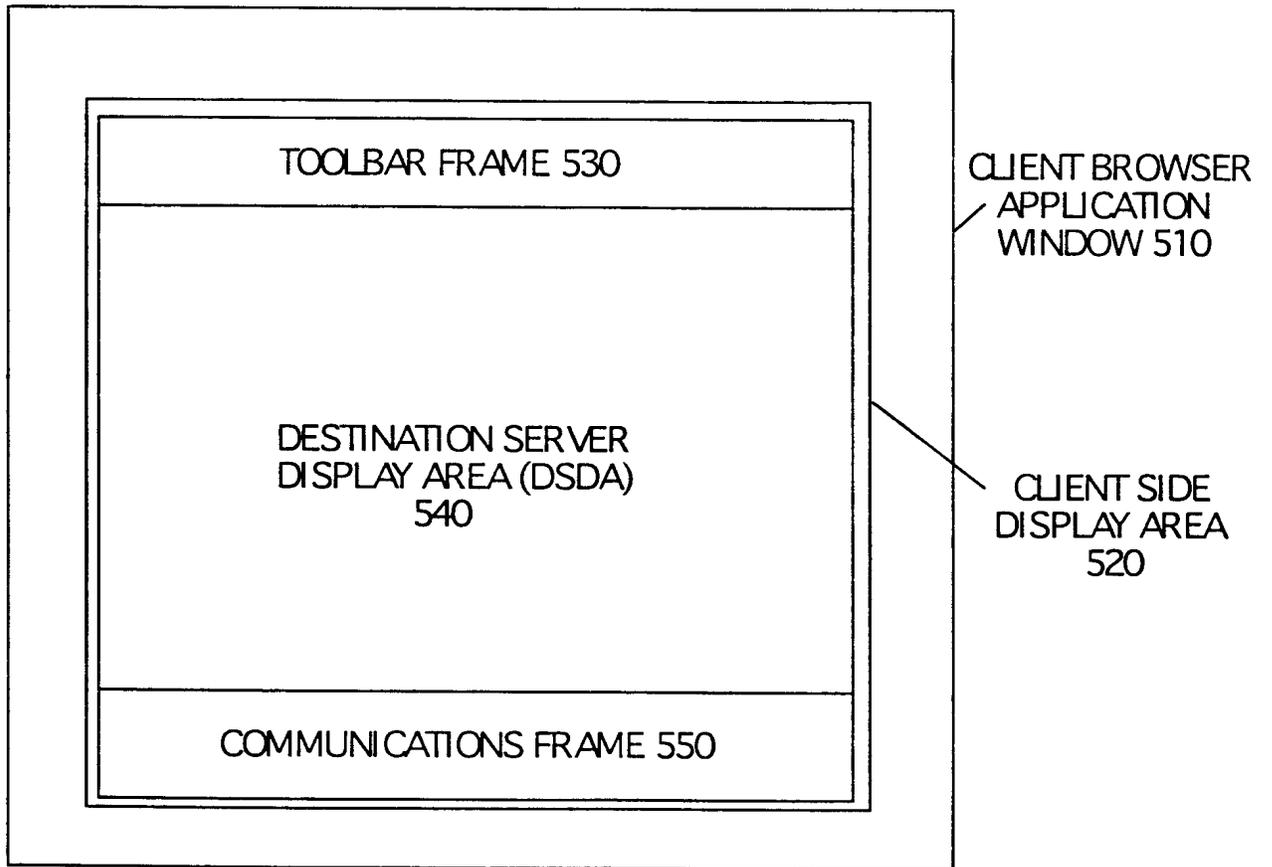
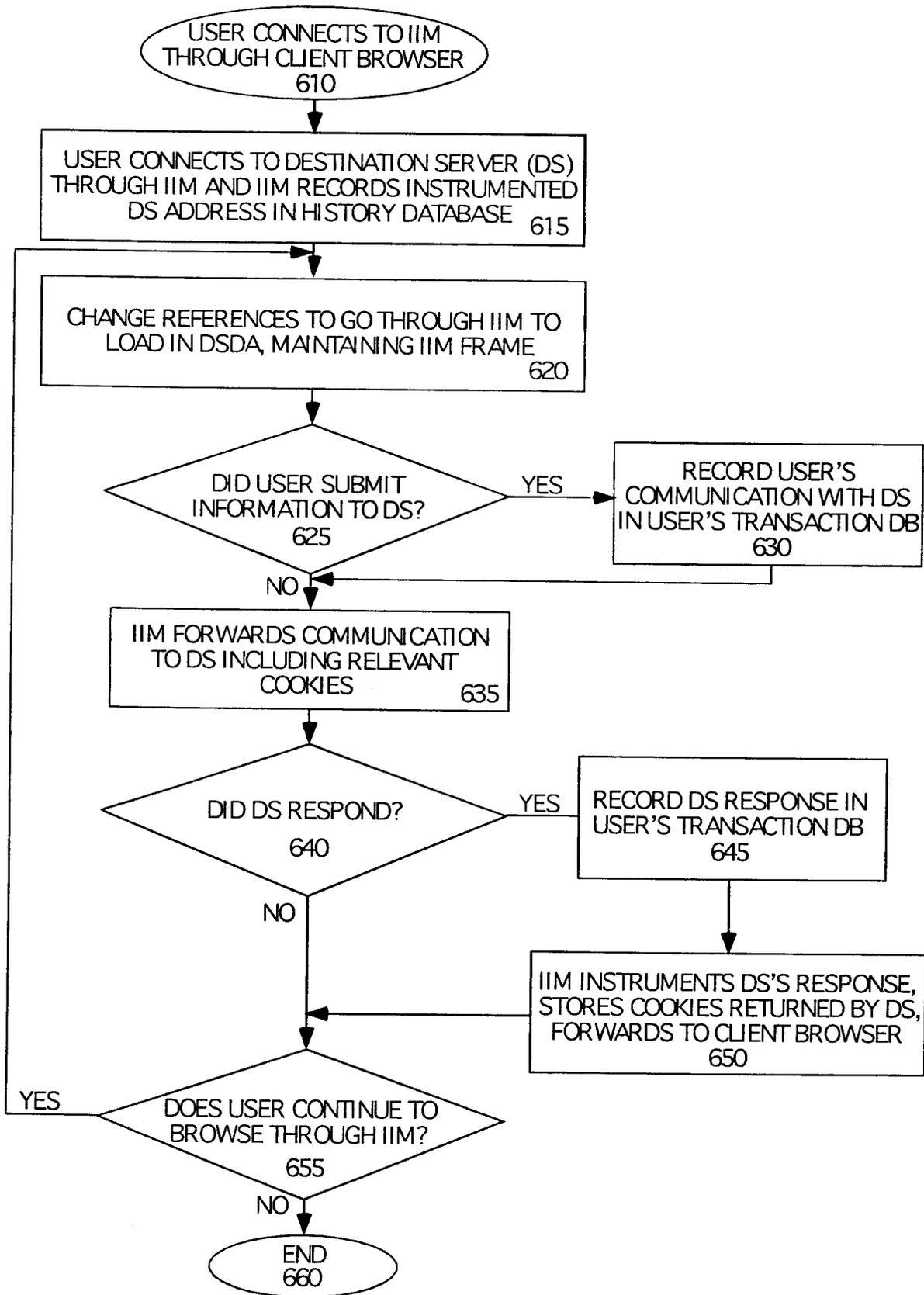


FIG. 5

7/25



8/25

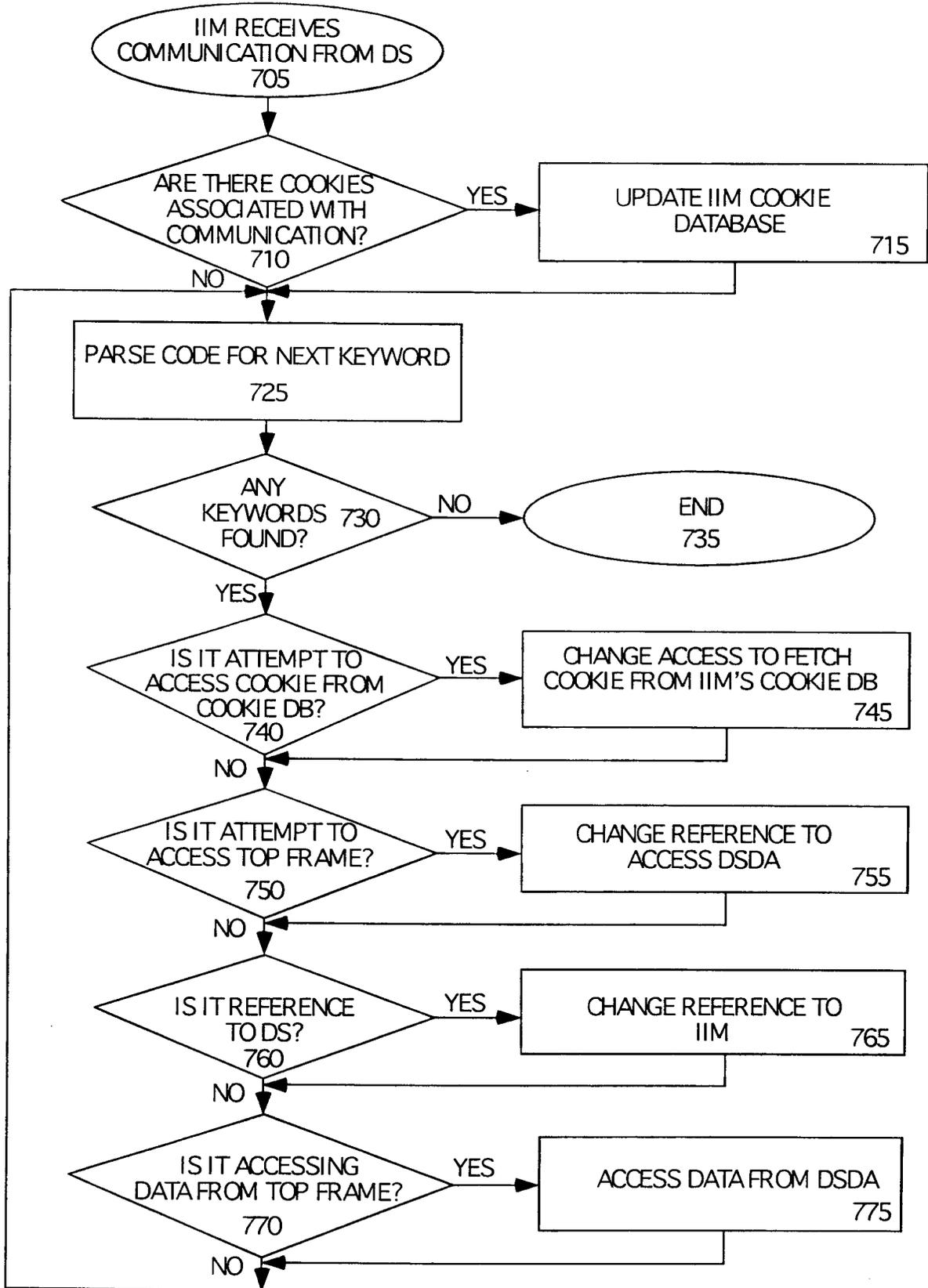


FIG. 7

9/25

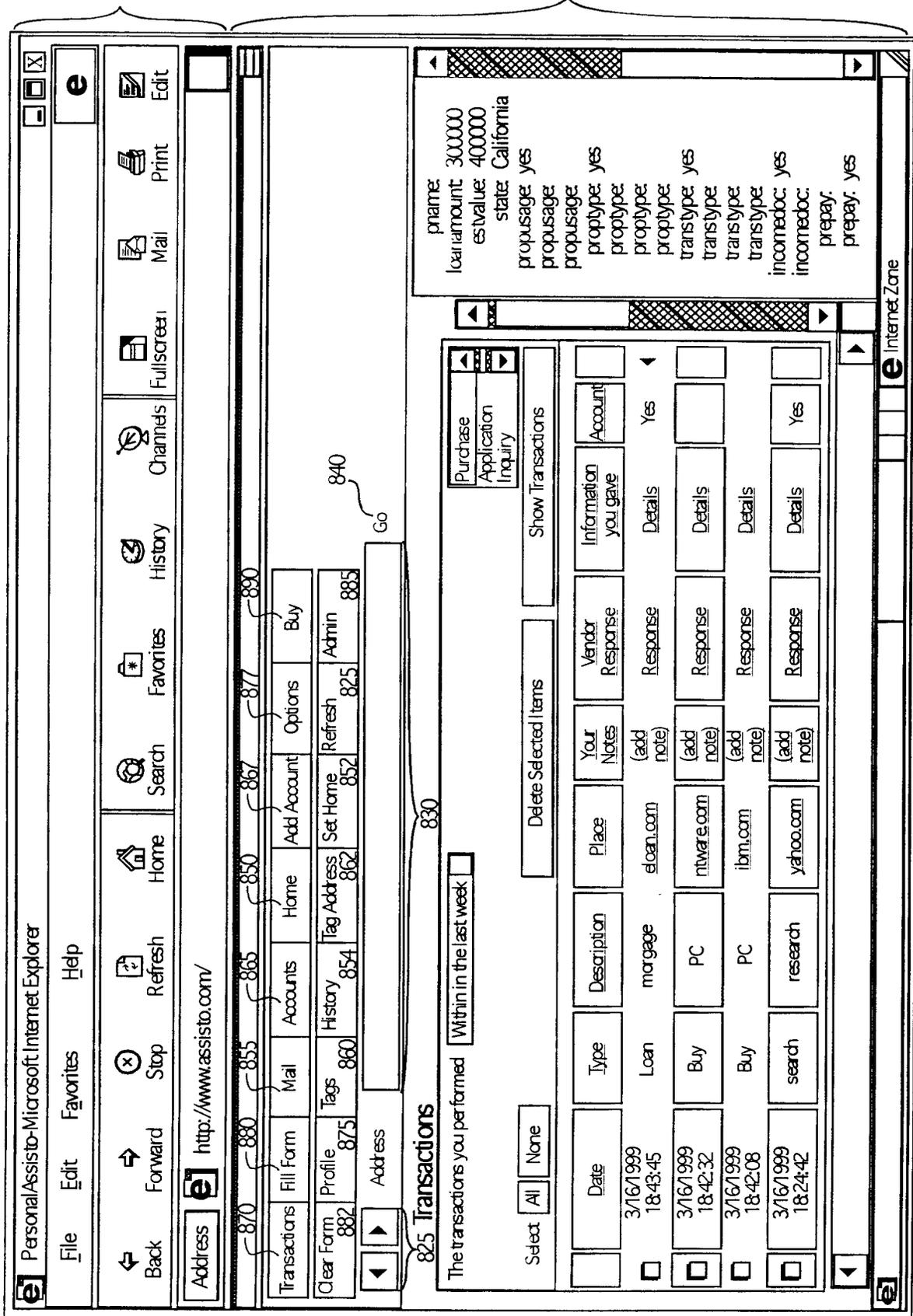
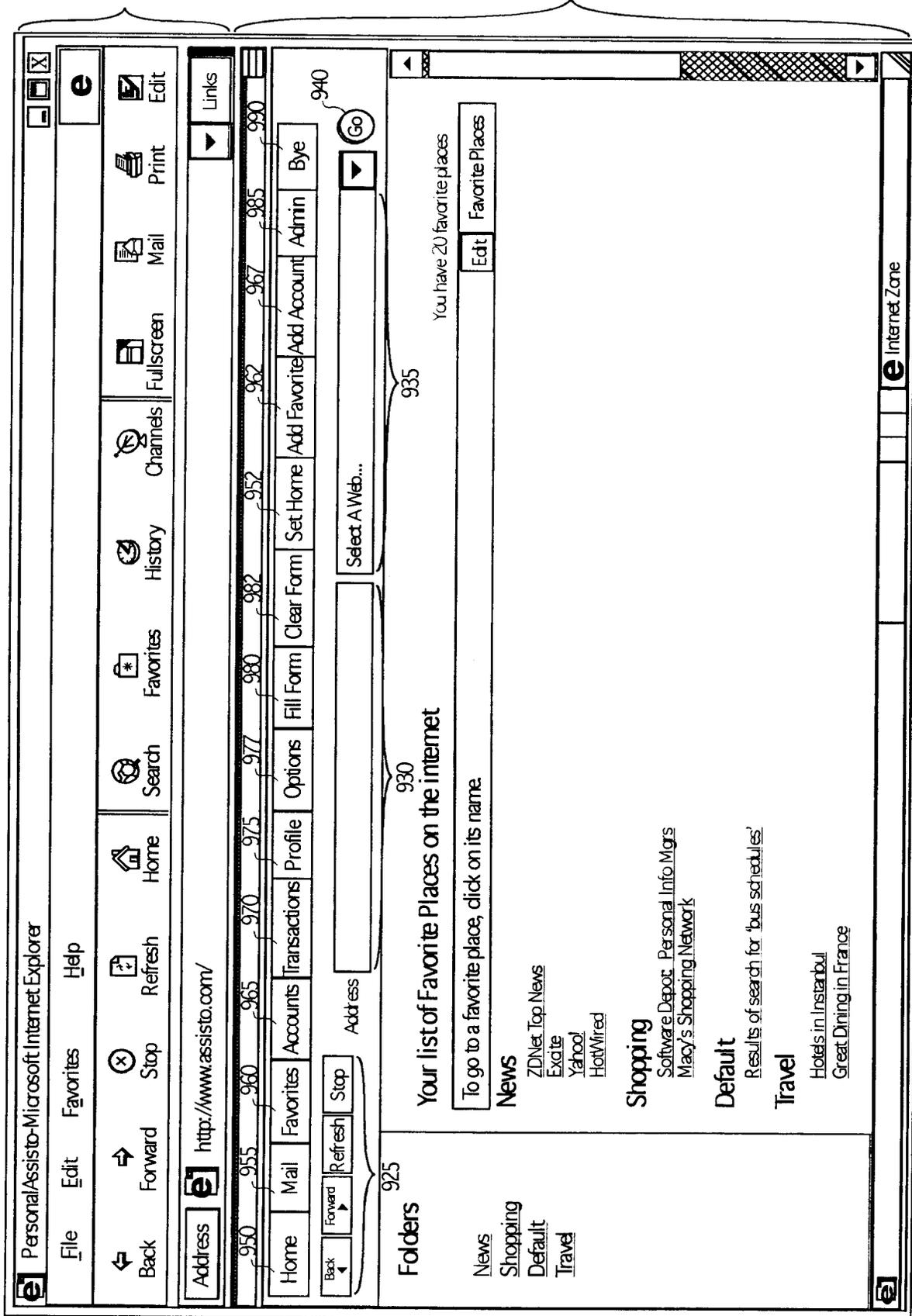


FIG. 8

10/25



920

920

945

FIG. 9

11/25

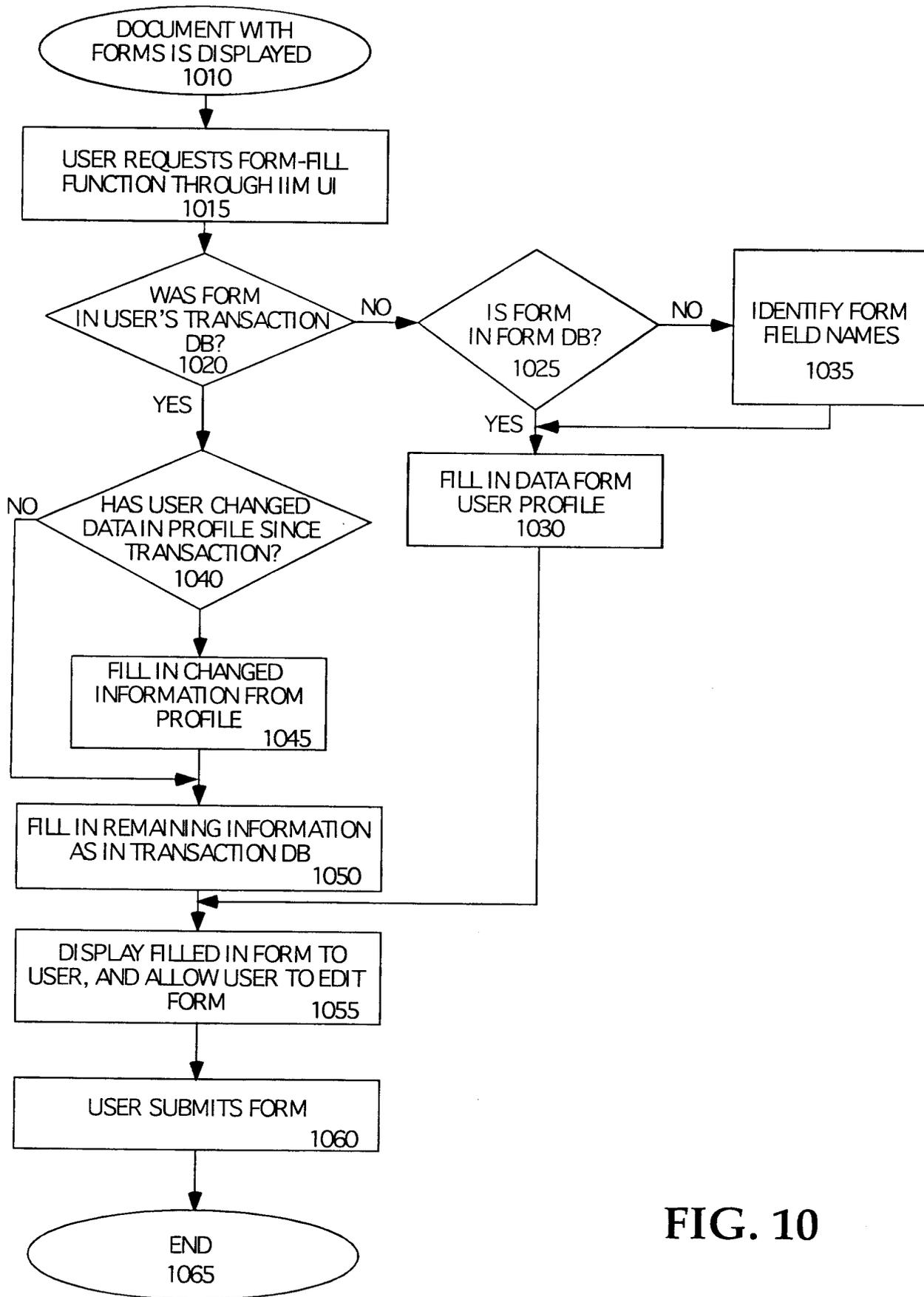


FIG. 10

12/25

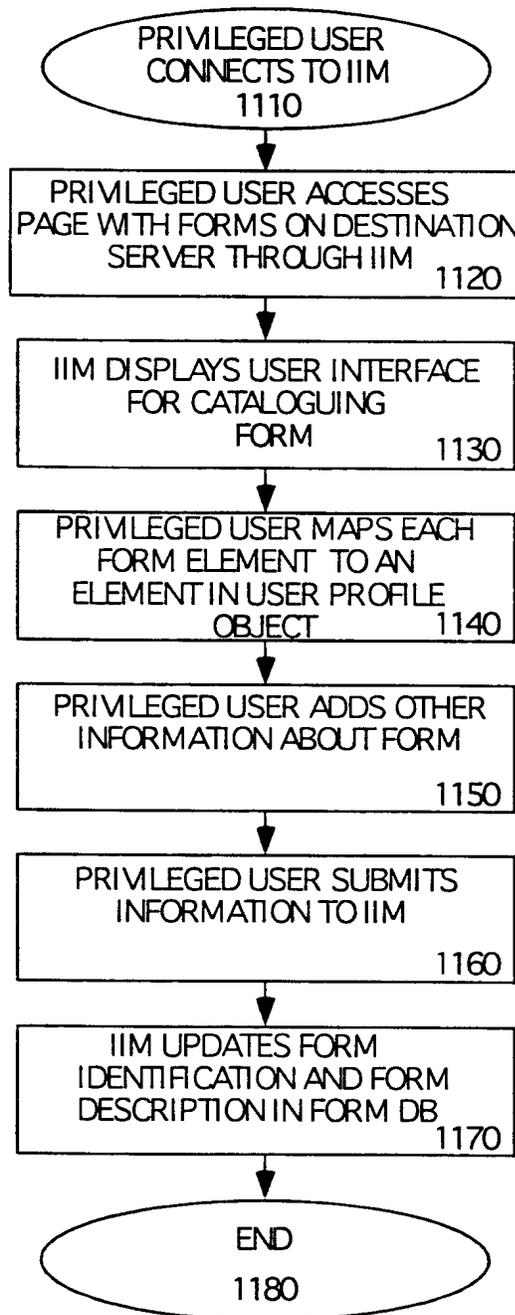


FIG. 11

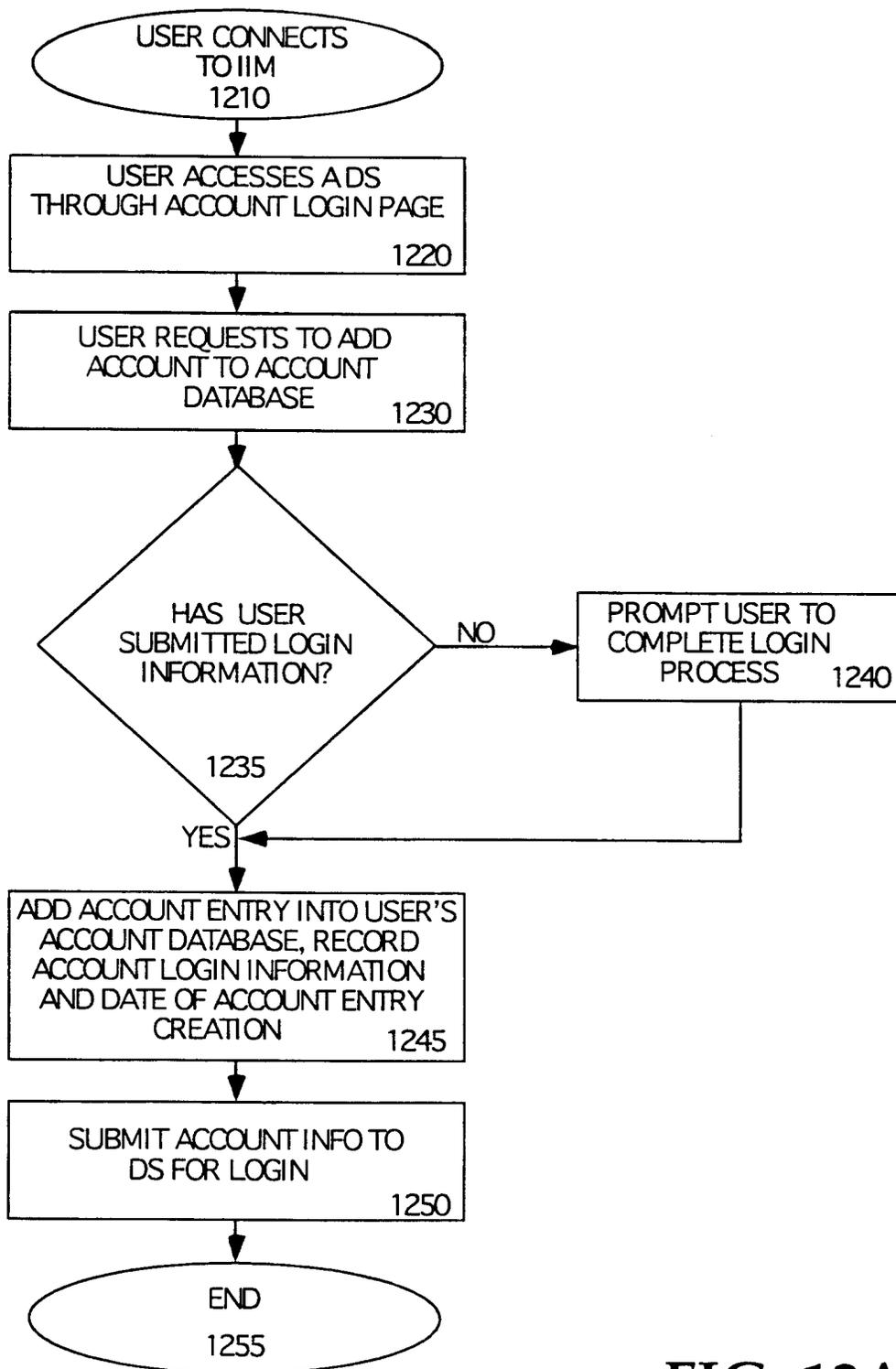


FIG. 12A

14/25

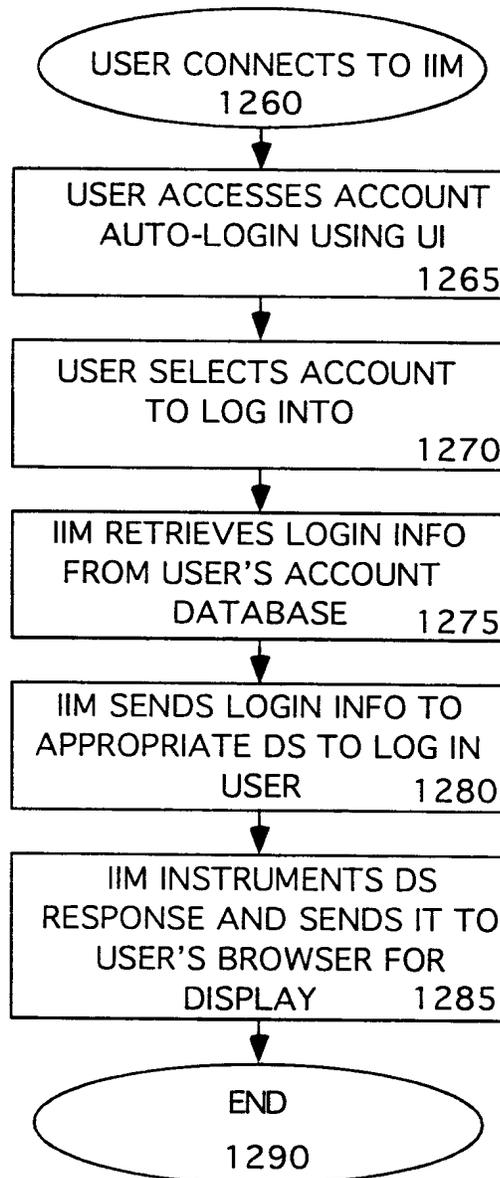


FIG. 12B

15/25

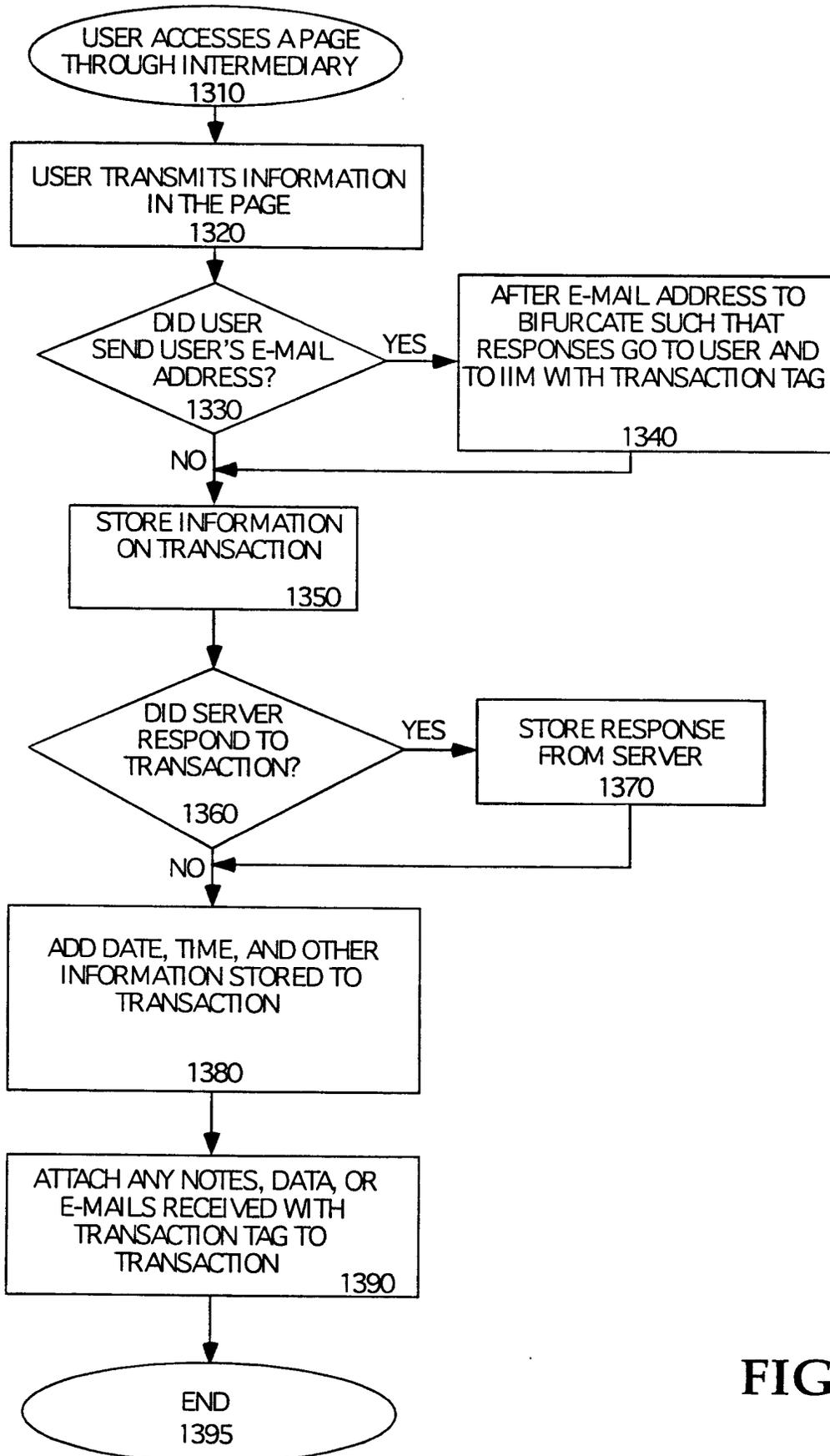


FIG. 13

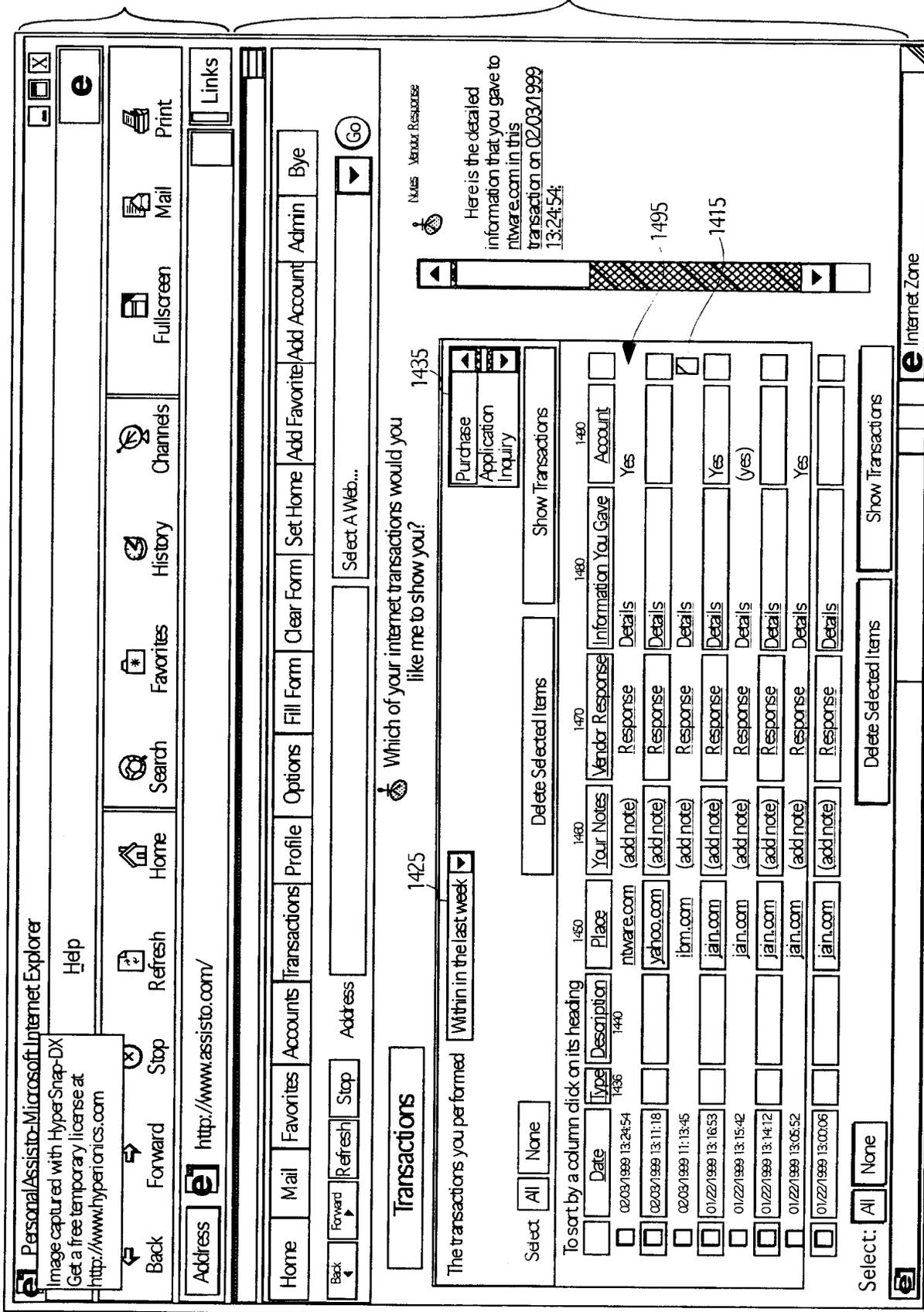


FIG. 14

17/25

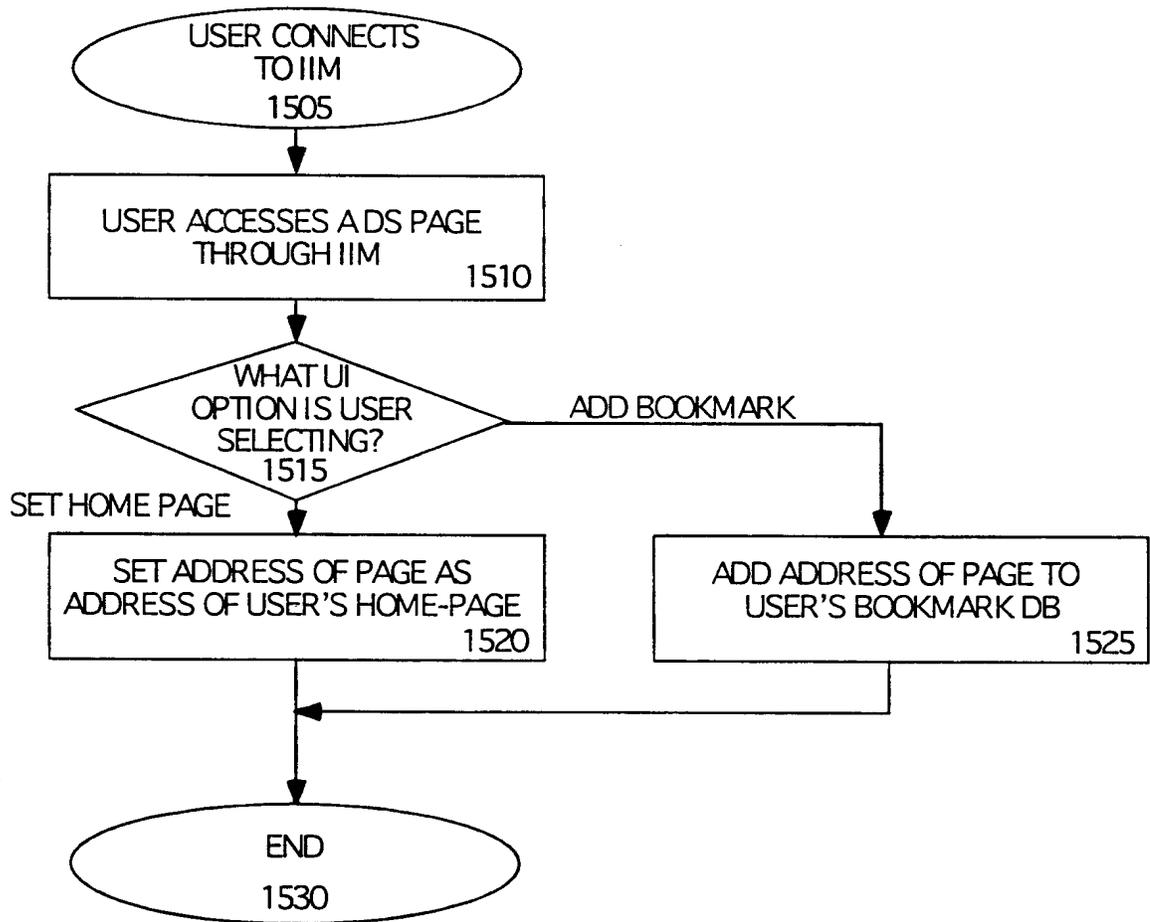


FIG. 15A

SUBSTITUTE SHEET (RULE 26)

18/25

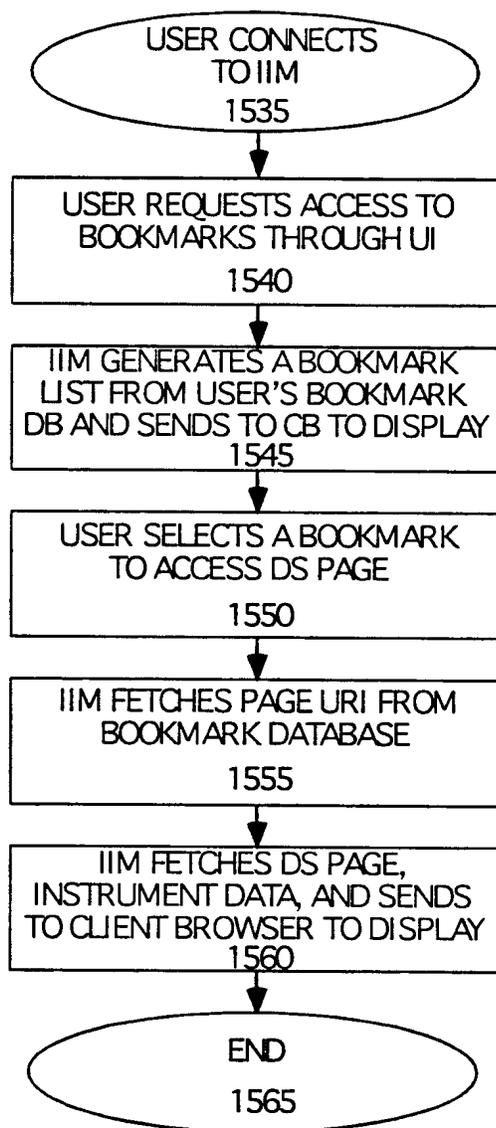


FIG. 15B

19/25

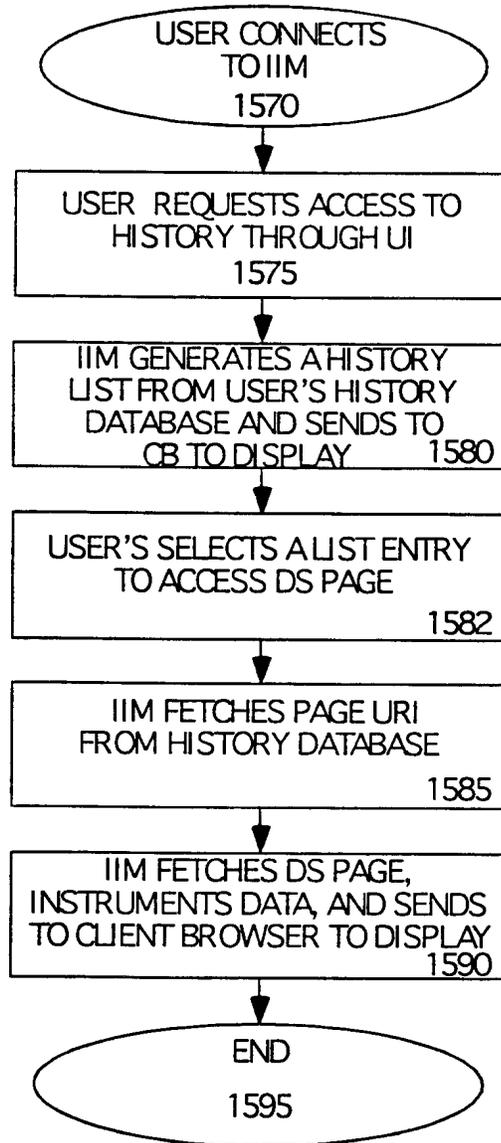


FIG. 15C

20/25

Original Code	Altered Code	Comments
HTML		
<code><base href="anyURL"></code>	<pre> <base href="http://www.DS.com/myDo cument.html"> base href="anyURL"> </pre>	<p>www.DS.com is the hostname of the DS. The IIM inserts the first <base> tag line after the <HTML> tag and before the <HEAD> tag, and before any existing <base> tags</p>
<code><form action="/actionURL"></code>	<pre> SaveOrigAction(form, actionURL) <form action=" http://www.IIM.com/redirect?act =http://www.DS.com /actionURL"> </pre>	<p>www.IIM.com is the hostname of the IIM and www.DS.com is the hostname of the DS. SaveOrigAction() is a Javascript function that saves the form's original action.</p>
<code><applet codebase="/codebase" code="applet.class"></code>	<pre> <applet codebase=" http://www.IIM.com/redirect?cb = http://www.DS.com/codebase" code="applet.class"> </pre>	<p>www.IIM.com is the hostname of the IIM and www.DS.com is the hostname of the DS.</p>
<code><frame src="/myFrame.html"></code> other tags e.g., <script>, <area>, <layer>, 	<pre> <frame src="http://www.IIM.com/redire ct?src=http://www.DS.com/myF rame.html"> </pre>	<p>www.IIM.com is the hostname of the IIM and www.DS.com is the hostname of the DS.</p>
Javascript		
<code>link.href="newLocation"</code>	<pre> setURLProperty (link, "href", "newLocation") </pre>	<p>setURLProperty() sets the value of the property href to the value "http://www.IIM.com/redirect?url=http://www.DS.com/newLocation", where www.IIM.com is the hostname of the IIM and www.DS.com is the hostname of the DS.</p>
<code>link.onclick = originalOnClick</code>	<pre> function addNewLinkOndick(link){ link.onclickOrig = link.onclick; link.onclick = newLinkOndick; } function newLinkOndick(link){ if (link.originalHref == null) link.originalHref = link.href; var newHref = getFullPathName(link.originalHref); link.href = http://www.IIM.com/redirect?url =newHref; return link.onclickOrig(); } </pre>	<p>getFullPathName() returns the full pathname URL of the HTML and www.IIM.com is the hostname of the IIM. The function addNewLinkOndick() is called when the HTML document is first loaded</p>

FIG. 16A

Original Code	Altered Code	Comments
document.write(StringToWrite)	writeDocument (document, StringToWrite);	writeDocument() recursively modifies all HTTP control points that occur in StringToWrite
window.open(newLocation);	openWindow (window, newLocation);	openWindow calls windows.open() with the argument "http://www.IS.com/redirect?url=http://www.DS.com/newLocation"
form.onsubmit = originalOnSubmit	<pre>function addNewFormOnsubmit(form){ form.onsubmitOrig = form.onsubmit; form.onsubmit = newFormOnsubmit; } function newFormOnsubmit (form) { if (form.originalAction == null) {form.originalAction = form.action; } var newAction = getFullPathName(form.original Action); form.action = http://www.IIM.com/redirect?url =newAction; return form.onsubmitOrig(); }</pre>	getFullPathName() returns the full pathname URL of the HTML document and www.IIM.com is the hostname of the IIM. The function addNewFormOnsubmit() is called when the HTML document is first loaded
Java		
class.java.net.Socket	Extends java.net.Socket and overrides various constructors.	The extended method modifies the host and port arguments. The modified host argument is the hostname of the IIM. the modified port argument is the port of the IIM
java.applet.AppletContext	Extends java.applet.AppletContext and overrides various constructors	The extended method modifies the url argument. The modified url sends the HTTP request to the IIM with the full pathname of the original url as a query parameter
class java.applet.Applet	Extends java.applet.Applet and overrides various constructors	The extended method modifies the url argument. The modified url sends the HTTP request to the IIM with the full pathname of the original url as query parameter

FIG. 16B

Original Code	Altered Code	Comments
HTTP Headers		
referer, e.g. value = origDoc.htm	referer, http://www.DS.com/origDoc.htm	referer value is replaced with the full pathname of the document's original URL
content-type, value = null	content-hype, e.g. value = image/gif	If the value of the content-type is null, the IIM sets this header to a value that describes the type of content contained in the document
refresh, e.g. 5000; origDoc.htm	5000; http://www.IS.com/redirect?ref= http://www.DS.com/origDoc.htm	IIM replaces the URL portion of the value of the refresh header with the full pathname of the document's original URL
301, 302 status codes, e.g., URI value = http://www.DS.com/origDoc.html	301, 302 status codes, URI value = http://www.IS.com/redirect?uri= http://www.DS.com/origDoc.html	www.IIM.com is the hostname of the IIM
201, 303, 305, 307 status codes URI values	201, 303, 305, 307 status codes modified URI values	See 301, 302 status codes

FIG. 16C

Original Code	Altered Code	Comments
HTML		
target = "_top" for <a href>, <frame>, <form> and <base> tags	target = "DSFrameName"	where DSFrameName is the name of the DSDA top frame.
target = "_parent" for <a href>, <frame>, <form> and <base> tags	target = "DSFrameName"	where DSFrameName is the name of the DSDA top frame, if the current window evaluates to the DSDA top frame.
Javascript		
top.locationProperty = value	setTopProperty (currentWindow, locationProperty, value)	setTopProperty sets a location property on the DSDA top frame with name locationProperty to have value value
window.parent.locationProperty = value	setTopProperty (window.parent, locationProperty, value)	setTopProperty sets a location property on the DSDA top frame with name locationProperty to have value value if window.parent evaluates to the top or IIM frame
Java		
java.applet.AppletContext.showDocument(url, target)	<pre> newShowDocument (window, url, target) { java.applet.AppletContext.showDocument(url, newTarget) } </pre>	newShowDocument calls java.applet.AppletContext.showDocument where newTarget is the name of the DSDA frame if target equals "_top" or if target equals "_parent" and window is the DSDA frame, "newTarget" is set to "target" otherwise

FIG. 17

Original Code	Altered Code	Comments
Javascript		
String = document.cookie	string = getCookie(window, document)	getCookie() gets the cookie's value from the IIM and assigns the value to string
document.cookie = cookieString	setCookie(window, document, cookieString)	setCookie() sends the value of cookieString to the IIM to be managed
HTTP Headers		
	cookie	IIM sends this header to the DS on a need basis
Java		
javax.servlet.http.Cookie	<pre>public class ISCookie implements Serializable { private long_creationTime; public boolean hasExpired(){ //expiration function code } }</pre>	maintains the cookie's creation time and contains convenience routines that determine if the cookie has expired
Cookies	<pre>public class Cookies extends PersistantObject implements Serializable { public String getCookie (URL url) {} public static Cookie parseCookie(String str) {} public void addCookie(ISCookie coo, URL url) {} private boolean validCookie(ISCookie coo, URL url) {} }</pre>	extends the Java persistent object class, and saves in a user database the user-specific cookie information that the DS sent in the set-cookie header. This class also finds the cookies in the cookie database that are valid for a certain URI based on well known Cookie rules and returns a Cookie string for a given URI.

FIG. 18

Original Code	Altered Code	Comments
Javascript		
window.location = newLocation	setLocation(currentWindow, window, newLocation)	setLocation() sets the window.location to the value "http://www.IIM.com/redirect?url=http://www.DS.com/newLocation", where www.IIM.com is the hostname of the IIM and www.DS.com is the hostname of the DS.
saveLocation = window.location	saveLocation = getLocation(currentWindow, window)	where getLocation returns window.location except when window is equal to the top frame, then it returns the DSDA top top frame's location
top.userProperty = value	setTopProperty (currentWindow, userProperty, value)	where setTopProperty sets a user property on the DSDA top frame with name userProperty to have value value
window.parent.userProperty = value	setTopProperty (window.parent, userProperty, value)	where setTopProperty sets a user property on the DSDA top frame with name userProperty to have value value if window.parent evaluates to the top frame.

FIG. 19